2011-03

On algorithms for nonlinear minimax and min-max-min problems and their efficiency

Pee, Eng Yau

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

ON ALGORITHMS FOR NONLINEAR MINIMAX AND MIN-MAX-MIN PROBLEMS AND THEIR EFFICIENCY

by

Pee Eng Yau

March 2011

Dissertation Supervisor: Johannes O. Royset

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2011	Dissertation

4. TITLE AND SUBTITLE On Algorithms for Nonlinear Minimax and Min-Max-Min Problems and Their Efficiency	5. FUNDING NUMBERS
6. AUTHOR Pee Eng Yau	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N.A.

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT(*maximum 200 words*)

This dissertation approaches the solution of optimization models with uncertain parameters by considering the worst-case value of the uncertain parameters during optimization. We consider three problems resulting from this approach: a finite minimax problem (FMX), a semi-infinite minimax problem (SMX), and a semi-infinite min-max-min problem (MXM). In all problems, we consider nonlinear functions with continuous variables. We find that smoothing algorithms for (FMX) may only have sublinear rates of convergence, but their complexity in the number of functions is competitive with other algorithms. We present two new smoothing algorithms with novel precision-adjustment schemes for (FMX). For (SMX) algorithms, we present a novel way of expressing rate of convergence in terms of computational work instead of the typical number of iterations, and show how the new way allows for a fairer comparison of different algorithms. We propose a new approach to solve (MXM), based on discretization and reformulation of (MXM) as a constrained finite minimax problem. Our approach is the first to solve (MXM) in the general case where the innermost feasible region depends on the variables in the outer problems. We conduct numerical studies for all three problems.

14. SUBJECT TERMS Finite and semi-infinite minimax, Generalized min-max-min, Discretization, Rate of convergence, Complexity	15. NUMBER OF PAGES 172
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ON ALGORITHMS FOR NONLINEAR MINIMAX AND MIN-MAX-MIN PROBLEMS AND THEIR EFFICIENCY

Pee Eng Yau
Principal Analyst, Singapore Defence Science and Technology Agency
B.Eng., National University of Singapore, 1996
M.S., Naval Postgraduate School, 2002

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

from the
NAVAL POSTGRADUATE SCHOOL
March 2011

Author:

_____
Pee Eng Yau

Approved by:

Johannes O. Royset Assistant Professor of Operations Research Dissertation Supervisor	Gerald G. Brown Distinguished Professor of Operations Research
R. Kevin Wood Distinguished Professor of Operations Research	W. Matthew Carlyle Associate Professor of Operations Research
Craig W. Rasmussen Associate Professor of Applied Mathematics	

Approved by:

_____
Robert F. Dell, Chairman, Department of Operations Research

Approved by:

_____
Douglas Moses, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This dissertation approaches the solution of optimization models with uncertain parameters by considering the worst-case value of the uncertain parameters during optimization. We consider three problems resulting from this approach: a finite minimax problem (FMX), a semi-infinite minimax problem (SMX), and a semi-infinite min-max-min problem (MXM). In all problems, we consider nonlinear functions with continuous variables. We find that smoothing algorithms for (FMX) may only have sublinear rates of convergence, but their complexity in the number of functions is competitive with other algorithms. We present two new smoothing algorithms with novel precision-adjustment schemes for (FMX). For (SMX) algorithms, we present a novel way of expressing rate of convergence in terms of computational work instead of the typical number of iterations, and show how the new way allows for a fairer comparison of different algorithms. We propose a new approach to solve (MXM), based on discretization and reformulation of (MXM) as a constrained finite minimax problem. Our approach is the first to solve (MXM) in the general case where the innermost feasible region depends on the variables in the outer problems. We conduct numerical studies for all three problems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1

A. MOTIVATION AND BACKGROUND 1

B. SCOPE OF DISSERTATION 2

C. CONTRIBUTIONS 4

D. MATHEMATICAL BACKGROUND 5

1. Continuity of Max Functions 5

2. Rate of Convergence 6

3. Consistent Approximations 7

E. ORGANIZATION 10

II. FINITE MINIMAX PROBLEM 11

A. INTRODUCTION 11

B. EXPONENTIAL SMOOTHING 15

C. RATE OF CONVERGENCE AND COMPLEXITY 17

1. Ill-Conditioning of Smoothed Problem 19

2. Complexity 22

3. Optimal Parameter Choice 25

4. Rate of Convergence 27

D. SMOOTHING ALGORITHMS AND ADAPTIVE PRECISION ADJUSTMENT 31

1. Smoothing Algorithm Based on Optimality Function . . 32

2. Smoothing Algorithm Using Cost Descent 36

3. Complexity 45

E. NUMERICAL RESULTS 46

1. Selection of a Robust ϵ for Active-Set Algorithms 48

2. Comparison 51

F. CONCLUSIONS FOR FINITE MINIMAX 57

III. SEMI-INFINITE MINIMAX PROBLEM 59
A. INTRODUCTION 59
B. EFFICIENCY OF DISCRETIZATION ALGORITHM 63
1. Discretization 63
2. Ideal Algorithm Map 68
3. Adaptive Discretization Algorithm 69
4. Quadratically Convergent Algorithm Map 71
5. Linearly Convergent Algorithm Map 74
6. Sublinearly Convergent Algorithm Map 77
7. Smoothing Algorithm Map 80
C. EFFICIENCY OF ϵ-SUBGRADIENT METHOD 86
D. NUMERICAL RESULTS 98
1. Problem Instance of Uncertainty Dimension One 100
2. Problem Instance of Uncertainty Dimension Two 102
3. Problem Instance of Uncertainty Dimension Three 102
E. CONCLUSIONS FOR SEMI-INFINITE MINIMAX 105
IV. SEMI-INFINITE MIN-MAX-MIN PROBLEM 107
A. INTRODUCTION 107
B. DEFENDER-ATTACKER-DEFENDER EXAMPLE 110
C. APPROACH TO SOLVE THE MIN-MAX-MIN PROBLEM . . 114
1. Constructing a Finite Minimax Problem 114
2. Algorithm for Semi-Infinite Min-Max-Min 120
D. NUMERICAL RESULTS 121
E. CONCLUSIONS FOR SEMI-INFINITE MIN-MAX-MIN 123
V. CONCLUSIONS AND FUTURE RESEARCH 125
A. CONCLUSIONS 125
B. FUTURE RESEARCH 127
APPENDIX A. FINITE MINIMAX PROBLEMS 129

APPENDIX B. FINITE MINIMAX ALGORITHM DETAILS AND PARAMETERS 133
APPENDIX C. SEMI-INFINITE MINIMAX PROBLEMS 135
APPENDIX D. SEMI-INFINITE MINIMAX ALGORITHM DETAILS AND PARAMETERS 139
APPENDIX E. SEMI-INFINITE MIN-MAX-MIN PROBLEM PARAMETERS AND RESULTS 141
LIST OF REFERENCES 145
INITIAL DISTRIBUTION LIST 151

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

1. Smoothed Problems. 16
2. Optimal Supply and Flow Solution for (SMXM). 122
3. Optimal Supply and Flow Solution for (GMXM). 124

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

1. Run times based on ϵ for ϵ-PPP. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems. 49
2. Run times based on ϵ for SQP-2QP. 49
3. Run times based on ϵ for SMQN and Algorithm II.2. 50
4. Run times based on ϵ for Algorithm II.3. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems. 51
5. Run times (in seconds) for various algorithms. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (II.92) after six hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 10^{-3}$ at termination. 53
6. Similar results as in Table 5, but with larger q. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (II.92) after six hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 0.01$ at termination. 55
7. Run times (in seconds) of algorithms on problem instance ProbN. "SD" and "QN" indicate that Algorithm II.3 uses $B_{p\Omega}(\cdot)$ given by (II.47) and (II.48), respectively. The word "mem" indicates that the algorithm terminates due to insufficient memory. 56
8. Run times (in seconds) for SProbA using Algorithm III.1 with ϵ-PPP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours. The word "mem" means that the algorithm terminates due to insufficient memory. 101
9. Run times (in seconds) for SProbA using Algorithm III.1 with SQP-2QP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance. The word "mem" means that the algorithm terminates due to insufficient memory. 102
10. Run times (in seconds) for SProbA using Algorithm III.3. The numbers in parentheses indicate the number of iterations. 102

11. Run times (in seconds) for SProbB using Algorithm III.1 with ϵ-PPP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours. 103
12. Run times (in seconds) for SProbB using Algorithm III.1 with SQP-2QP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance. 103
13. Run times (in seconds) for SProbB using Algorithm III.3. The numbers in parentheses indicate the number of iterations. 103
14. Run times (in seconds) for SProbC using Algorithm III.1 with ϵ-PPP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours. 104
15. Run times (in seconds) for SProbC using Algorithm III.1 with SQP-2QP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours. 105
16. Run times (in seconds) for SProbC using Algorithm III.3. The numbers in parentheses indicate the number of iterations. 105
17. Finite minimax problem instances. An asterisk * indicates that the problem instance are created by the authors. 131
18. Semi-infinite minimax problem instances. The starting value y_0 is only relevant for the ϵ-subgradient algorithm, Algorithm III.3. 137
19. Random generators used to produce the DAD problem parameters in Table 20. The phrase 5×U(2,4), 5×U(10,13) represents that a total of ten random numbers are generated, the first five are uniformly distributed between two and four, and the last five numbers are uniformly distributed between ten and 13. 141
20. Defender-Attacker-Defender Network Data. 143
21. Results for (SMXM). 143
22. Results for (GMXM). 143

ACKNOWLEDGMENTS

I thank my advisor, Dr. Johannes Royset, for his invaluable advice and constant guidance throughout this arduous but fruitful journey. I am grateful to my other committee members, Dr. Gerald Brown, Dr. Kevin Wood, Dr. Matt Carlyle, and Dr. Craig Rasmussen, for the knowledge that they have imparted me in their classes, and also for their careful and detailed review of the dissertation.

My office-mates, Hiro Sato, Ali Al-Rowaei, Dave Ruth, Jay Foraker, Anthony Tvaryanas, Mumtaz Karatas, and Helcio Vieira Junior, have made this journey much easier and more enjoyable. I will miss the late nights studying in the office with Hiro, making coffee with Dave in the morning, and chatting with Jay on issues from consistent approximations to iPads.

I am grateful to my bosses at the Defence Science and Technology Agency for giving me this wonderful opportunity to pursue my PhD at NPS. Special mention to my ex-boss, Mr. Koh Wee Liam, who supported my PhD scholarship application and made all this possible.

I want to thank my mum, Poa Choo, and my sisters, Suat Hoon, Suat Keng, and Suat Hong, for their care and support. Touring the U.S. with them during their visits has been a real highlight for my stay here.

I want to thank my wife, Puay Joo, for taking good care of our kids, Verdell and Jerrall, so that I can concentrate all my efforts on my research, for her constant encouragement, for her understanding when we have to remain in Monterey during quarterly breaks because of my research work, and for her superb soups. To Verdell and Jerrall, now that this dissertation is complete, I can finally stay home after dinner to play Wii with you. LET'S GO Mario and Yellow Mushroom...

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Optimization problems with uncertain parameters arise in numerous applications. One possible approach to handle such problems is to consider the worst-case value of the uncertain parameter during optimization. We consider three problems resulting from this approach: a finite minimax problem (FMX), a semi-infinite minimax problem (SMX), and a semi-infinite min-max-min problem (MXM). In all problems, we consider nonlinear functions with continuous variables. We develop rate of convergence and complexity results, and propose algorithms for solving these optimization problems.

In (FMX), we solve a minimax problem with a finite number of variables and functions. We develop rate of convergence and complexity results of smoothing algorithms for solving (FMX) with many functions. We find that smoothing algorithms may only have sublinear rates of convergence, but their complexity in the number of functions is competitive with other algorithms due to small computational work per iteration. We present two smoothing algorithms with novel precision-adjustment schemes and carry out a comprehensive numerical comparison with other algorithms from the literature. We find that the proposed algorithms are competitive with SQP algorithms, and especially efficient for problem instances with many variables, or where a significant number of functions are nearly active at stationary points.

The numerical results also indicate that smoothing with first-order gradient methods is likely the only viable approach to solve (FMX) with a large number of functions and variables due to memory issues.

For (SMX), we solve a minimax problem with a finite number of variables, but an infinite number of functions. We develop and compare the rate of convergence results for various fixed and adaptive discretization algorithms, as well as an ϵ-subgradient algorithm. We present a novel way of expressing rate of convergence, in terms of computational work instead of the typical number of iterations. Hence, we

are able to identify algorithms that are competitive due to low computational work per iteration even if they require many iterations. We show that a fixed discretization algorithm with a quadratically or linearly convergent algorithm map for the discretized problem can achieve the same asymptotic convergence rate attained by an adaptive discretization algorithm. We show that under certain convexity assumptions, the rates of convergence for discretization algorithms depend on the dimension of the uncertain parameters, while the rates of convergence for ϵ-subgradient algorithms are independent of the dimension of the uncertain parameters under certain convexity-concavity assumptions. This indicates that under convexity-concavity assumptions, discretization algorithms are not competitive with ϵ-subgradient algorithms for problems with large dimensions of the uncertain parameters, and that conclusion is validated by our numerical results.

In (MXM), the variables in each layer of the problem vary within compact continuous sets. We consider two cases depending whether the inner feasible region is a constant set, which we denote by (SMXM), or depends on decision variables of the outer min-max problem, which we call the generalized semi-infinite min-max-min problem, and denote by (GMXM). We propose a new approach to solve (MXM), based on discretization and reformulation of (MXM) into a constrained finite minimax problem with a larger dimensionality than the original (MXM). Our approach is the first to solve (GMXM) in the literature and it also solves (SMXM). We apply our approach on a defender-attacker-defender network interdiction problem, which demonstrates the viability of the approach.

I. INTRODUCTION

A. MOTIVATION AND BACKGROUND

Most, if not all, decisions in the real world are made under some uncertainty, for example, Apple needs to decide on the plant capacity for manufacturing the iPad before knowing the demands for it, the Department of Homeland Security needs to make investment and operational decisions not knowing where and how the next terrorist attack will occur, and almost everyone invests in stocks and bonds not knowing if they will turn out to be profitable.

In optimization models, uncertainty usually shows up as uncertain parameters in the model formulation. A common approach taken to handle the uncertainty is to use the average value of the parameter, or to use its most-likely value, and then use deterministic optimization to find an optimal solution. There are many examples that show that optimal solutions based on such point estimates of uncertain parameters are not robust, i.e., small changes to the parameters cause the previously optimal solution to have a much worse outcome.

The importance of considering uncertainty in optimization can be seen by the number of techniques developed for it (Sahinidis, 2004; Rockafellar, 2007), among which are the tools of stochastic programming (Shapiro, Dentcheva, & Ruszczyński, 2009) and stochastic dynamic programming (Powell, 2007). The main challenge with the technique is the availability or even the existence of the probability distribution for certain parameters.

An example where no probability distribution exists is that of an adversarial situation, where an adversary wants to maximize damage to you, or minimize your ability to achieve certain objectives. In such problems it is reasonable to use a minimax formulation, to minimize the worst-case damage that can be caused by the adversary. Optimizing our actions against the worst-case scenario is the topic of this dissertation.

B. SCOPE OF DISSERTATION

In this dissertation, we consider three problems of increasing difficulty: an unconstrained finite minimax problem (FMX), an unconstrained semi-infinite minimax problem (SMX), and a constrained semi-infinite min-max-min problem (MXM). Specifically, we develop rate of convergence and complexity results, as well as algorithms for solving these problems. In all problems, we consider nonlinear functions with continuous variables.

In (FMX), we solve a minimax problem with a finite number of variables and functions. There are several approaches to solve (FMX). We consider one such approach, that of smoothing algorithms. In smoothing algorithms (see for example Polak, Royset, & Womersley, 2003; Polak, Womersley, & Yin, 2008; Ye, Liu, Zhou, & Liu, 2008; Li, 1992; Xu, 2001), we create a smooth function that approximates the non-differentiable pointwise maximum function and minimize the smooth approximating function. As noted in Polak et al. (2003), the key strength of smoothing algorithms is that they convert minimax problems into simple, smooth, and unconstrained optimization problems that can be solved using any standard unconstrained optimization algorithms. While complexity and rate of convergence have been studied extensively for nonlinear programs and minimax problems (see for example Nemirovski & Yudin, 1983; Drezner, 1987; Wiest & Polak, 1991; Nesterov, 1995; Ariyawansa & Jiang, 2000; Nesterov & Vial, 2004; Nesterov, 2004), the topics have been largely overlooked in the specific context of smoothing algorithms for (FMX). We discuss complexity and rate of convergence for smoothing algorithms for (FMX), and propose two new smoothing algorithms to solve (FMX). We consider problem instances of (FMX) with up to 10,000,000 functions and up to 10,000 variables in the numerical studies to compare the new smoothing algorithms with other algorithms from the literature.

For (SMX), we solve a minimax problem with a finite number of variables, but an infinite number of functions. The focus of our research for (SMX) is on a novel way of expressing rate of convergence of algorithms. Consider two (SMX) al-

gorithms, a linearly convergent algorithm and a superlinearly convergent algorithm. Since conventional rate of convergence do not consider computational work, it is possible for the linearly convergent algorithm to generate an iterate every second, while the superlinearly convergent algorithm to generate an iterate every hour. Or worse still, the superlinearly convergent algorithm takes an hour to generate the first iterate, and the run time to generate subsequent iterates doubles at every iteration. As mentioned, this lack of correlation between rate of convergence and run time is because conventional rate of convergence do not consider computational work. We propose a new way of expressing rate of convergence, which considers computational work. We select several (SMX) algorithms to illustrate how the new way of expressing rate of convergence addresses the issues of the conventional way described above. Specifically, we examine discretization and ϵ-subgradient algorithms. Discretization algorithms are one of the more popular classes of algorithms for solving SIPs due to their simplicity. In discretization algorithms, we solve a sequence of finite minimax problems, where the number of functions considered increases. Since the computational work to solve a finite minimax problem depends on the number of functions in the problem, a discretization algorithm takes increasingly longer time to generate an iterate as the discretization algorithm progresses. An ϵ-subgradient algorithm does not use discretization to solve (SMX) and is well-known to have a sublinear rate of convergence. Its run time does not vary much between iterations. Compared to the conventional way of expressing rate of convergence, we show that the new way allows us to conduct a fairer comparison between the ϵ-subgradient algorithm and discretization algorithms. We also conduct numerical studies to validate the rate-of-convergence results that we obtain.

In (MXM), the variables in each layer of the problem vary within a compact set with uncountable cardinality. We consider two cases depending whether the inner feasible region is a constant set, which we denote by (SMXM), or depends on decision variables of the outer min-max problem, which we call the generalized semi-infinite

min-max-min problem, and denote by (GMXM). The problem (MXM) is difficult to solve, which explains the rather limited literature on (SMXM), and so far, there is no solution approach for (GMXM). We propose a new approach to solve (MXM), based on discretization and reformulation of (MXM) into a constrained finite minimax problem. We apply the approach on a defender-attacker-defender network interdiction problem for a 10-node 18-arc network to demonstrate the viability of the approach.

C. CONTRIBUTIONS

The main contributions of this dissertation are as follows. We provide the first complexity and rate-of-convergence analyses of smoothing algorithms for solving (FMX). We develop two new smoothing algorithms with novel precision-adjustment schemes. We conduct a comprehensive numerical comparison of our algorithms with other algorithms from the literature, considering problem instances with the number of functions two orders of magnitude larger than problem instances considered in the literature. The numerical results indicate that the two new smoothing algorithms are competitive with the other algorithms compared.

For (SMX), we present a novel way of expressing rate of convergence, in terms of computational work instead of the typical number of iterations, which allows for a fairer comparison of algorithms. We show that a fixed discretization algorithm with quadratically or linearly convergent algorithm map can achieve the same asymptotic convergence rate in terms of computational work as the one attained by an adaptive discretization algorithm. We show that under certain convexity-concavity assumptions, discretization algorithms are not competitive with ϵ-subgradient algorithms for problems with large dimension of the uncertain parameters, which we also validated in numerical tests.

We develop the first exact algorithm for (GMXM), which also results in a novel approach for solving (SMXM). If (MXM) has an objective function that is convex in the inner and outer minimization variables, and the inner and outer feasible regions are convex, then our algorithm guarantees convergence to a global minimizer of (MXM).

D. MATHEMATICAL BACKGROUND

This section defines notation and mathematical concepts used throughout this dissertation. Throughout the dissertation, $\mathbb{R}^n$ denotes the n-dimensional Euclidean space, $\mathbb{N} \stackrel{\triangle}{=} \{1, 2, ...\}$, $\mathbb{N}_0 \stackrel{\triangle}{=} \mathbb{N} \cup \{0\}$, $|\cdot|$ represents the cardinality operator, $\|\cdot\|$ represents the Euclidean norm operator, A^T denotes the transpose of the matrix A, and $x_i \to^K x$ represents that given a $K \subset \mathbb{N}$, for every $\epsilon > 0$, there exists an $i_1 \in K$ such that $|x_i - x| \leq \epsilon$ for all $i \geq i_1, i \in K$. Other than the above notation, which is used to denote the same quantities throughout this dissertation, some notation may be used to represent different quantities in different chapters.

1. Continuity of Max Functions

The following results on the continuity of the pointwise maximum (applies to minimum as well) function are used repeatedly throughout the dissertation.

Proposition I.1. *Suppose that the functions $f^j : \mathbb{R}^d \to \mathbb{R}, j \in Q \stackrel{\triangle}{=} \{1, 2, ..., q\}, q \in \mathbb{N}$, are continuous for all $x \in \mathbb{R}^d, d \in \mathbb{N}$. Then the pointwise maximum function $\psi : \mathbb{R}^d \to \mathbb{R}$ defined by*

$$\psi(x) \stackrel{\triangle}{=} \max_{j \in Q} f^j(x) \tag{I.1}$$

is continuous for all $x \in \mathbb{R}^d$. □

Proposition I.2. *Let $Y \subset \mathbb{R}^m$ be a compact set, and the functions $\phi(\cdot, y)$, where $\phi : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$, be continuous for all $y \in Y$ on $\mathbb{R}^d$, Then the pointwise maximum function $\psi : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ defined by*

$$\psi(x) \stackrel{\triangle}{=} \max_{y \in Y} \phi(x, y) \tag{I.2}$$

is continuous for all $x \in \mathbb{R}^d$. □

The proofs for Propositions I.1 and I.2 can be found on pp. 51 and 187 of Demyanov and Malozamov (1974), repectively.

2. Rate of Convergence

Two key performance measures of an optimization algorithm are its complexity and rate of convergence. We define the different rates of convergence next, based on Bertsekas (1999, pp. 63-65) and Nocedal and Wright (2006, pp. 619-620).

Consider a sequence of points $\{x_n\}_{n=0}^{\infty} \subset \mathbb{R}^d$ converging to $x^* \in \mathbb{R}^d$. Rate of convergence can be evaluated using an error function $e_n : \mathbb{R}^d \to \mathbb{R}$, where $e_n \geq 0$ for all $n \in \mathbb{N}_0$ and $e_n \to 0$ as $n \to \infty$. The two common-used error functions are based on Euclidean distance

$$e_n = \|x_n - x^*\|, \tag{I.3}$$

and function values

$$e_n = |f(x_n) - f(x^*)|. \tag{I.4}$$

We say that the convergence is sublinear if

$$\limsup_{n\to\infty} \frac{e_{n+1}}{e_n} = 1. \tag{I.5}$$

The convergence is linear if there exist $c \in (0, 1)$ and $n_1 \in \mathbb{N}_0$ such that

$$\frac{e_{n+1}}{e_n} \leq c, \tag{I.6}$$

for all $n \geq n_1$. Convergence is superlinear if

$$\limsup_{n\to\infty} \frac{e_{n+1}}{e_n} = 0. \tag{I.7}$$

We say that we have order of convergence $r > 1$ if there exist a $c > 0$ and a $n_1 \in \mathbb{N}_0$ such that

$$\frac{e_{n+1}}{(e_n)^r} \leq c, \tag{I.8}$$

for all $n \geq n_1$. When $r = 2$, we call the convergence quadratic.

If two sequences converge sublinearly, we say that they have the same rate, even if the constants are different. Similar comments hold for linear and superlinear convergence. We say that two sequences that are both superlinear converge at the same rate. We say that two sequences that are both superlinear but with different orders converge at the same rate but with different orders. We also say that superlinear convergence is faster than linear convergence, which again is faster than sublinear convergence.

We say that an algorithm map used to solve a problem (P) converges sublinearly, linearly, or superlinearly if the sequence generated by the algorithm map converges sublinearly, linearly, or superlinearly, respectively.

3. Consistent Approximations

This subsection discusses the theory of consistent approximations (Polak, 2003, 1997). Consider the problem

$$(\mathrm{P}) \quad \min_{x \in X} f(x), \tag{I.9}$$

where $X \subset \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}$ is continuous.

Next, given $N \in \mathbb{N}$, consider an approximate problem to (P)

$$(\mathrm{P}_N) \quad \min_{x \in X_N} f_N(x), \tag{I.10}$$

where $X_N \subset \mathbb{R}^d$ and $f_N : \mathbb{R}^d \to \mathbb{R}$ is continuous.

Two properties are required for the approximating problems (as $N \to \infty$) to be consistent approximations to (P). First, we need the epi-convergence of (P_N) to (P) as $N \to \infty$. For a detailed discussion of epi-convergence; see Polak (1997, Section 3.3.1) or Rockafellar and Wets (1998, Sections 1B, 4B, & 7B). We here give essential definitions and results for our study.

We define the epigraph

$$E \stackrel{\triangle}{=} \left\{(x, z) \in \mathbb{R}^{d+1} \mid x \in X, z \geq f(x)\right\}. \tag{I.11}$$

The set E consists of all the points in $\mathbb{R}^{d+1}$ on or above the function $f(\cdot)$. Similarly, the epigraph

$$E_N \stackrel{\triangle}{=} \left\{(x,z) \in \mathbb{R}^{d+1} \mid x \in X_N, z \geq f_N(x)\right\}. \tag{I.12}$$

Epi-convergence of (P_N) to (P) as $N \to \infty$ is then defined as set convergence of the epigraphs E_N to E, in the sense of Painlevé-Kuratowski, as in Definition 5.3.6 of Polak (1997). For completeness, we restate the definition of set convergence in the sense of Painlevé-Kuratowski.

Definition I.1. Consider a sequence of sets $\{A_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$.

(i) We define the distance between a point $\hat{x}$ and a set A_i as

$$\rho(\hat{x}, A_i) \stackrel{\triangle}{=} \inf\{\|x - \hat{x}\| \mid x \in A_i\}. \tag{I.13}$$

The point $\hat{x}$ is a *limit point* of $\{A_i\}_{i=0}^{\infty}$ if $\rho(\hat{x}, A_i) \to 0$ as $i \to \infty$ (that is, $\hat{x}$ is a limit point of $\{A_i\}_{i=0}^{\infty}$ if there exist a $x_i \in A_i$ for all $i \in \mathbb{N}$, such that $x_i \to \hat{x}$, as $i \to \infty$).

(ii) The point $\hat{x}$ is a *cluster point* of $\{A_i\}_{i=0}^{\infty}$ if it is a limit point of a subsequence of $\{A_i\}_{i=0}^{\infty}$.

(iii) We denote the set of limit points of $\{A_i\}_{i=0}^{\infty}$ by $\liminf A_i$, and we denote the set of cluster points of $\{A_i\}_{i=0}^{\infty}$ by $\limsup A_i$.

(iv) The sets A_i converge in the sense of Painlevé-Kuratowski to the set A as $i \to \infty$ if $\liminf A_i = \limsup A_i = A$. □

An alternate way to prove epi-convergence is provided by the following proposition, extracted from Polak (2003, Theorem 3.1).

Proposition I.3. *The sequence of problems* $\{(\mathrm{P}_N)\}_{N\in\mathbb{N}}$ *epi-converges to* (P) *as* $N \to \infty$ *if and only if*

(i) for every $x \in X$*, there exists a sequence* $\{x_N\}_{N\in\mathbb{N}}$*, where* $x_N \in X_N$*,* $x_N \to x$ *as* $N \to \infty$*, and* $\limsup f_N(x_N) \leq f(x)$*;*

(ii) for every infinite sequence $\{x_N\}_{N\in K}$*, where* $K \subset \mathbb{N}$*,* $x_N \in X_N$ *for all* $N \in K$*, and* $x_N \to^K x$ *as* $N \to \infty$*, then* $x \in X$ *and* $\liminf f_N(x_N) \geq f(x)$*.* □

The importance of epi-convergence is stated in the next result, extracted from Polak (2003, Theorem 3.2).

Proposition I.4. *Suppose that the sequence of problems* $\{(\mathrm{P}_N)\}_{N\in\mathbb{N}}$ *epi-converges to* (P) *as* $N \to \infty$. *Then the following facts hold:*

(i) If $\{\hat{x}_N\}$ *is a sequence of global minimizers of* (P_N) *and there exists an infinite subset* $K \in \mathbb{N}$ *such that* $\hat{x}_N \to^K \hat{x}$ *as* $N \to \infty$, *then* $\hat{x}$ *is a global minimizer of* (P), *and* $f_N(\hat{x}_N) \to^K f(\hat{x})$ *as* $N \to \infty$.

(ii) If $\{\hat{x}_N\}$ *is a sequence of local minimizers of* (P_N) *sharing a common radius of attraction* $\rho > 0$ *(i.e., for all* $N \in \mathbb{N}$, $f_N(\hat{x}_N) \leq f_N(x)$ *for all* $x \in X_N$ *such that* $\|x - \hat{x}_N\| \leq \rho$*), and there exists an infinite subset* $K \in \mathbb{N}$ *such that* $\hat{x}_N \to^K \hat{x}$ *as* $N \to \infty$, *then* $\hat{x}$ *is a local minimizer of* (P), *and* $f_N(\hat{x}_N) \to^K f(\hat{x})$ *as* $N \to \infty$. □

Epi-convergence does not rule out the possibility that an arbitrary sequence of local minimizers of (P_N) may have an accumulation point that is neither a local minimizer nor a stationary point. To ensure that accumulation points of a sequence of stationary points of (P_N) are stationary points of (P), a suitable characterization of stationarity is required, such as the use of optimality functions as defined by Definition 3.3 of Polak (2003).

Definition I.2. A function $\theta : \mathbb{R}^d \to \mathbb{R}$ is an *optimality function* for (P) if (i) $\theta(\cdot)$ is upper semi-continuous, (ii) $\theta(x) \leq 0$ for all $x \in \mathbb{R}^d$, and (iii) if $\hat{x}$ is a local minimizer of (P), then $\theta(\hat{x}) = 0$. Similarly, a function $\theta_N : \mathbb{R}^d \to \mathbb{R}$ is an optimality function for (P_N) if (i) $\theta_N(\cdot)$ is upper semi-continuous, (ii) $\theta_N(x) \leq 0$ for all $x \in \mathbb{R}^d$, and (iii) if $\hat{x}_N$ is a local minimizer of (P_N), then $\theta_N(\hat{x}_N) = 0$. □

We next define consistent approximations, as per Definition 3.4 of Polak (2003).

Definition I.3. The pairs $((\mathrm{P}_N), \theta_N(\cdot))$, in the sequence $\{((\mathrm{P}_N), \theta_N(\cdot))\}$ are *consistent approximations* to the pair $((\mathrm{P}), \theta(\cdot))$ if (i) (P_N) epi-converges to (P) as $N \to \infty$ and (ii) for any infinite sequence $\{x_N\}_{N\in K}, K \subset \mathbb{N}$ where $x_N \to x$, $\limsup \theta_N(x_N) \leq \theta(x)$. □

Consistent approximations ensure that given a sequence of approximate stationary points $\{x_N\}$, where $\theta_N(x_N) \to 0$ as $N \to \infty$, and $x_N \to \hat{x}$ as $N \to \infty$, then $\theta(\hat{x}) = 0$, i.e., $\hat{x}$ is a stationary point of (P).

E. ORGANIZATION

The remainder of this dissertation is outlined as follows. Chapter II develops results for rate of convergence and complexity for smoothing algorithms to solve (FMX). We present two new smoothing algorithms with novel precision-adjustment schemes and carry out a comprehensive numerical comparison with other algorithms from the literature. In Chapter III, we present a novel way of expressing rate of convergence, in terms of computational work instead of the typical number of iterations. We develop and compare rate-of-convergence results for various fixed and adaptive discretization algorithms as well as an ϵ-subgradient algorithm. In Chapter IV, we propose a new approach to solve (MXM). We apply the approach to solve a defender-attacker-defender network interdiction problem to illustrate the viability of our new approach. Chapter V covers the conclusions and future research opportunities.

II. FINITE MINIMAX PROBLEM

A. INTRODUCTION

This chapter considers finite minimax problems of the form

$$\text{(FMX)} \quad \min_{x \in \mathbb{R}^d} \psi(x), \tag{II.1}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\psi(x) \stackrel{\triangle}{=} \max_{j \in Q} f^j(x), \tag{II.2}$$

and $f^j : \mathbb{R}^d \to \mathbb{R}$, $j \in Q \stackrel{\triangle}{=} \{1, 2, ..., q\}$, $q \in \mathbb{N}$, are twice continuously differentiable. (FMX) is "finite" as we consider a finite number of functions, as compared to the "semi-infinite" problems (SMX) and (MXM) where we consider an infinite number of functions. Finite minimax problems of the form (FMX) may occur in engineering design (Polak, 1987), control system design (Polak, Salcudean, & Mayne, 1987), portfolio optimization (Cai, Teo, Yang, & Zhou, 2000), best polynomial approximation (Demyanov & Malozemov, 1974), or as subproblems in semi-infinite minimax algorithms (Panier & Tits, 1989). We focus on minimax problems with many functions, i.e., large q, which may result from finely discretized semi-infinite minimax problems or optimal control problems; see for example Panier and Tits (1989); Zhou and Tits (1996). We develop algorithms for such problems and analyze their efficiency. An abbreviated version of this chapter is published separately (Pee & Royset, 2010).

The non-differentiability of the objective function in (FMX) poses the main challenge for solving minimax problems, as standard unconstrained optimization algorithms do not apply directly. Many algorithms have been proposed to solve (FMX); see for example Zhou and Tits (1996); Polak et al. (2003); Obasanjo et al. (2010) and references therein. One approach is sequential quadratic programming (SQP), where (FMX) is first reformulated into the standard nonlinear constrained problem

$$\text{(FMX}') \quad \min_{(x,z) \in \mathbb{R}^{d+1}} \{z \mid f^j(x) - z \leq 0 \ \ \forall j \in Q\} \tag{II.3}$$

and then an SQP algorithm is applied to (FMX′), advantageously exploiting the special structure in the new formulation (Zhou & Tits, 1996; Zhu, Cai, & Jian, 2009). Other approaches also based on (FMX′) include interior point methods (Sturm & Zhang, 1995; Obasanjo et al., 2010; Luksan, Matonoha, & Vlcek, 2005) and conjugate gradient methods in conjunction with exact penalties and smoothing (Ye et al., 2008).

Due to its aggressive active-set strategy, the SQP algorithm in Zhou and Tits (1996) appears especially promising for problems with many sequentially-related functions (in the sense that the values taken by $f^j(\cdot)$ are typically close to the values taken by $f^{j+1}(\cdot)$), as in the case of finely discretized semi-infinite minimax problems. The SQP algorithm in Zhou and Tits (1996) needs to solve two quadratic programs (QPs) in each iteration. Recently, Zhu et al. (2009) propose an SQP algorithm that requires the solution of only one QP per iteration, yet this algorithm retains global convergence and superlinear rate of convergence as in the algorithm in Zhou and Tits (1996). Furthermore, the algorithm in Zhu et al. (2009) does not use an active-set strategy. At a point $x \in \mathbb{R}^d$, we call a function $f^j(\cdot), j \in Q$, *active* if $f^j(x) = \psi(x)$, and ϵ-*active* ($\epsilon > 0$) if $f^j(x) \geq \psi(x) - \epsilon$. In general, an active-set strategy only considers functions that are ϵ-active (and disregards the other functions) at the current iterate, and thus greatly reduces the number of function and gradient evaluations at each iteration of an algorithm. While the number of iterations needed to solve a problem to required precision may increase, the overall effect may be a reduction in the number of function and gradient evaluations, and that may translate into reduced computing times. For example, Polak et al. (2008) reports a 75% reduction in the number of gradient evaluations, and Zhou and Tits (1996) reports reductions in computing times with active-set strategies.

In smoothing algorithms (see for example Polak et al., 2003, 2008; Ye et al., 2008; Li, 1992; Xu, 2001), we create a smooth function (using exponential smoothing, to be discussed in Section II.B) that approximates the non-differentiable $\psi(\cdot)$ and minimize the smooth approximating function. We refer to the resulting problem

of minimizing the smooth approximating function as a *smoothed problem*. As the smoothed problem remains unconstrained, one can use any standard unconstrained optimization algorithm, such as the Armijo Gradient or Newton methods (Polak et al., 2003) or a Quasi-Newton method (Polak et al., 2008).

A fundamental challenge for smoothing algorithms is that the smoothed problem becomes increasingly ill-conditioned as the approximation becomes more accurate. Consequently, the use of smoothing techniques is complicated by the need to balance the accuracy of the approximation with problem ill-conditioning. The simplest smoothing algorithm creates an accurate smooth approximating function and solve it. This simple static scheme of constructing a single smoothed problem and solving it is highly sensitive to the choice of accuracy and has poor numerical performance (Polak et al., 2003). An attempt to address this challenge by using a sequence of smoothed problems was first made in Xu (2001), where a precision parameter that controls approximation accuracy is initially set to a pre-selected value and then doubled at each iteration. Effectively, in this open-loop scheme to precision adjustment, the algorithm approximately solves a sequence of gradually more accurate approximations. This open-loop scheme is sensitive to the multiplication factor (Polak et al., 2003).

Polak et al. (2003) propose an adaptive precision-parameter adjustment scheme that controls problem ill-conditioning by keeping a smoothing precision parameter small when far from a stationary solution, and increasing the parameter as a stationary solution is approached. Numerical results show that the scheme manages ill-conditioning better than static and open-loop schemes. The smoothing algorithms in Xu (2001) and Polak et al. (2003) do not incorporate any active-set strategy.

Using the adaptive precision-parameter adjustment scheme in Polak et al. (2003), Polak et al. (2008) presents an active-set strategy for smoothing algorithms that tackles (FMX) with large q. We note that the convergence result in Theorem 3.3 of Polak et al. (2008) may be slightly incorrect as it claims stationarity for all

accumulation points of a sequence constructed by the algorithm in Polak et al. (2008). However, the proof for Theorem 3.3 of Polak et al. (2008) relies on Polak et al. (2003), which guarantees stationarity for only a single accumulation point.

This chapter examines smoothing algorithms for (FMX) with large q from two angles. First, we discuss complexity and rate of convergence for such algorithms. We define *complexity* as the computational work of an algorithm on a serial machine to obtain a solution that is within a specified error tolerance of the optimal solution of a problem, expressed as a function of the sizes of a specific set of inputs for the problem. While complexity and rate of convergence have been studied extensively for nonlinear programs and minimax problems (see for example Nemirovski & Yudin, 1983; Drezner, 1987; Wiest & Polak, 1991; Nesterov, 1995; Ariyawansa & Jiang, 2000; Nesterov & Vial, 2004; Nesterov, 2004), the topics have been largely overlooked in the specific context of smoothing algorithms for (FMX). A challenge here is the increasing ill-conditioning of the smoothed problem as the smoothing precision improves. We quantify the degree of ill-conditioning and use this result to analyze complexity and rate of convergence. We find that the rate of convergence may be sublinear, but low computational work per iteration yields complexity, as a function of q, that is competitive with several other algorithms.

Second, we consider implementation and numerical performance of smoothing algorithms. A challenge here is to construct schemes for selecting the precision parameter that guarantee convergence to stationary points and perform well empirically. As discussed above, static and open-loop precision-parameter adjustment schemes result in poor numerical performance and, thus, we develop two adaptive schemes. In extensive tests against other algorithms, smoothing algorithms with the adaptive schemes are competitive, and especially so for problem with many variables, or where a significant number of functions are nearly active at stationary points.

B. EXPONENTIAL SMOOTHING

For ease of analysis of active-set strategies, we consider the problem

$$(\text{FMX}_\Omega) \quad \min_{x\in\mathbb{R}^d} \psi_\Omega(x), \tag{II.4}$$

where $\psi_\Omega(x) \triangleq \max_{j\in\Omega} f^j(x)$, and $\Omega \subseteq Q$. When $\Omega = Q$, (FMX_Q) is identical to (FMX). For simplicity of notation, we drop subscripts Q in several contexts below. Next, for any $\Omega \subseteq Q$ and for a parameter $p > 0$, we define a smoothed problem to (FMX_Ω) by

$$(\text{FMX}_{p\Omega}) \quad \min_{x\in\mathbb{R}^d} \psi_{p\Omega}(x), \tag{II.5}$$

where

$$\begin{aligned} \psi_{p\Omega}(x) &\triangleq \frac{1}{p}\log\left(\sum_{j\in\Omega}\exp\left(pf^j(x)\right)\right) \\ &= \psi_\Omega(x) + \frac{1}{p}\log\left(\sum_{j\in\Omega}\exp\left(p(f^j(x)-\psi_\Omega(x))\right)\right) \end{aligned} \tag{II.6}$$

is an exponential penalty function. We denote (FMX_{pQ}) by (FMX_p) for brevity. This smoothing technique was introduced in Kort and Bertsekas (1972) and used in Polak et al. (2003, 2008); Ye et al. (2008); Li (1992); Xu (2001). The exponential penalty function has been commonly used in smoothing algorithms as it preserves differentiability (as formalized in Proposition II.1) and convexity (Li & Fang, 1997).

We denote the set of active functions at $x \in \mathbb{R}^d$ by $\widehat{\Omega}(x) \triangleq \{j \in \Omega | f^j(x) = \psi_\Omega(x)\}$. Except as stated in Appendix A, we denote components of a vector by superscripts.

The parameter $p > 0$ is a smoothing precision parameter, where a larger p implies higher precision as illustrated in Figure 1 and formalized by Proposition II.1; see for example Polak et al. (2008). In Figure 1, $\Omega = \{1, 2, 3\}$ and the subscript "Ω" has been dropped from the notation. The numbers in the subscripts are p values.

Proposition II.1. *Suppose that $\Omega \subseteq Q$ and $p > 0$.*

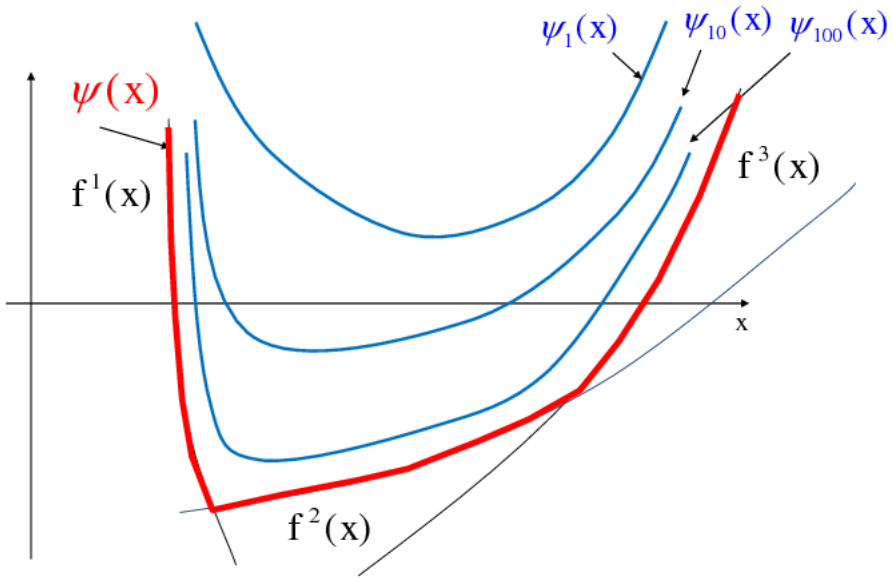

Figure 1. Smoothed Problems.

(i) *If the functions $f^j(\cdot)$, $j \in \Omega$, are continuous, then $\psi_{p\Omega}(\cdot)$ is continuous, and for any $x \in \mathbb{R}^d$, $\psi_{p\Omega}(x)$ decreases monotonically as p increases.*

(ii) *For any $x \in \mathbb{R}^d$,*

$$0 \leq \frac{\log|\widehat{\Omega}(x)|}{p} \leq \psi_{p\Omega}(x) - \psi_{\Omega}(x) \leq \frac{\log|\Omega|}{p}, \tag{II.7}$$

where $|\cdot|$ represents the cardinality operator.

(iii) *If the functions $f^j(\cdot)$, $j \in \Omega$, are continuously differentiable, then $\psi_{p\Omega}(\cdot)$ is continuously differentiable, with gradient*

$$\nabla\psi_{p\Omega}(x) = \sum_{j\in\Omega} \mu_p^j(x) \nabla f^j(x), \tag{II.8}$$

where

$$\mu_p^j(x) \triangleq \frac{\exp(pf^j(x))}{\sum_{k\in\Omega}\exp(pf^k(x))} = \frac{\exp(p[f^j(x) - \psi_{\Omega}(x)])}{\sum_{k\in\Omega}\exp(p[f^k(x) - \psi_{\Omega}(x)])} \in (0,1), \tag{II.9}$$

and $\sum_{j\in\Omega}\mu_p^j(x) = 1$ for all $x \in \mathbb{R}^d$.

(iv) If the functions $f^j(\cdot)$, $j \in \Omega$, are twice continuously differentiable, then $\psi_{p\Omega}(\cdot)$ is twice continuously differentiable, with Hessian

$$\begin{aligned}\nabla^2\psi_{p\Omega}(x) &= \sum_{j\in\Omega}\mu_p^j(x)\nabla^2 f^j(x) + p\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T \\ &\quad -p\left[\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\right]\left[\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\right]^T \qquad \text{(II.10)}\end{aligned}$$

for all $x \in \mathbb{R}^d$. □

We define a continuous, nonpositive optimality function $\theta_\Omega : \mathbb{R}^d \to \mathbb{R}$ for all $x \in \mathbb{R}^d$ by

$$\theta_\Omega(x) \stackrel{\triangle}{=} -\min_{\mu\in\Sigma_\Omega}\left\{\sum_{j\in\Omega}\mu^j(\psi_\Omega(x) - f^j(x)) + \tfrac{1}{2}\left\|\sum_{j\in\Omega}\mu^j\nabla f^j(x)\right\|^2\right\}, \qquad \text{(II.11)}$$

where $\Sigma_\Omega \stackrel{\triangle}{=} \{\mu \in \mathbb{R}^{|\Omega|} \mid \mu^j \geq 0 \ \forall j \in \Omega, \sum_{j\in\Omega}\mu^j = 1\}$. The following optimality condition for (FMX_Ω) is expressed in terms of $\theta_\Omega(\cdot)$; see Theorems 2.1.1, 2.1.3, and 2.1.6 of Polak (1997).

Proposition II.2. *Suppose that the functions $f^j(\cdot), j \in Q$, are continuously differentiable and that $\Omega \subseteq Q$. If $x^* \in \mathbb{R}^d$ is a local minimizer for* (FMX_Ω)*, then* $\theta_\Omega(x^*) = 0$. □

At stationary points of ($\text{FMX}_{p\Omega}$), the continuous, nonpositive optimality function $\theta_{p\Omega} : \mathbb{R}^d \to \mathbb{R}$ defined by $\theta_{p\Omega}(x) \stackrel{\triangle}{=} -\frac{1}{2}\|\nabla\psi_{p\Omega}(x)\|^2$ for all $x \in \mathbb{R}^d$, vanishes to zero.

C. RATE OF CONVERGENCE AND COMPLEXITY

This section examines the following basic smoothing algorithm, for which we develop a series of complexity and rate-of-convergence results. We use this simple algorithm to gain some fundamental insights on smoothing algorithms, but yet maintain tractability of the analysis. When they exist, we denote optimal solutions of

(FMX) and (FMX_p) by x^* and x_p^*, respectively, and the corresponding optimal values by ψ^* and ψ_p^*. The algorithm applies the Armijo Gradient Method to (FMX_p), starting at an initial point x_0. The value of p is fixed at p^* and it guarantees that Proposition II.3 stated below holds. The Armijo Gradient Method uses the steepest descent search direction and the Armijo stepsize rule to solve an unconstrained problem; see for example Algorithm 1.3.3 of Polak (1997).

Algorithm II.1. Smoothing Armijo Gradient Algorithm

Data: Error tolerance $t > 0, x_0 \in \mathbb{R}^d$.

Parameter: $\delta \in (0, 1)$.

Step 1. Set $p^* = (\log q)/((1-\delta)t)$.

Step 2. Generate a sequence $\{x_i\}_{i=0}^{\infty}$ by applying Armijo Gradient Method to (FMX_{p^*}). □

In this dissertation, we have several algorithms (including Algorithm II.1) with no termination criteria stated in the algorithm procedure. In general for nonlinear programming, there are often more than one possible termination criterion for each algorithm. For example, a possible termination criterion for unconstrained nonlinear optimization is the norm of the search direction falls below a certain small number. Determining an appropriate criterion is often application dependent. In all our numerical studies, we terminate the algorithms when (i) the current iterate falls within a certain error tolerance of the optimal solution or objective function value, or (ii) the solution satisfies the default tolerances of the solver used. We state the termination criterion in the numerical section of each chapter.

Algorithm II.1 has the following property.

Proposition II.3. *Suppose that $q \geq 2$ and Step 2 of Algorithm II.1 has generated a point $x_i \in \mathbb{R}^d$ such that $\psi_{p^*}(x_i) - \psi_{p^*}^* \leq \delta t$. Then $\psi(x_i) - \psi^* \leq t$.*

Proof. By the optimality of $\psi_{p^*}^*$ and (II.7), $\psi_{p^*}^* \leq \psi_{p^*}(x^*) \leq \psi^* + (\log q)/p^*$. Thus, $-\psi^* \leq -\psi_{p^*}^* + (\log q)/p^*$. Based on (II.7), $\psi(x_i) \leq \psi_{p^*}(x_i)$ and hence, $\psi(x_i) - \psi^* \leq$

$\psi_{p^*}(x_i) - \psi^*_{p^*} + (\log q)/p^*$. Since $\psi_{p^*}(x_i) - \psi^*_{p^*} \leq \delta t$ and p^* is as in Step 1, the conclusion follows. □

In the proposition above, the number of functions considered has been constrained to be two or more ($q \geq 2$) as it is not meaningful to take the pointwise maximum of a single function. For a fixed $p > 0$, the rate of convergence of the Armijo Gradient Method as applied to (FMX_p) is well known (see for example p. 60 of Polak, 1997). However, the value of the precision parameter p^* in Algorithm II.1 is dictated by q and t (see Step 1), which complicates the analysis. For large values of q or small values of t, p^* is large and hence (FMX_{p^*}) may be ill-conditioned as observed empirically (Polak et al., 2003). In this chapter, we quantify the ill-conditioning of (FMX_p) as a function of p and obtain complexity and rate of convergence results for Algorithm II.1.

1. Ill-Conditioning of Smoothed Problem

The following strong convexity assumption is a standard assumption required for complexity and rate of convergence analyses.

Assumption II.4. *The functions $f^j(\cdot), j \in \mathbb{N}$, are*

(i) twice continuously differentiable and

(ii) there exists an $m > 0$ such that

$$m\|y\|^2 \leq \langle y, \nabla^2 f^j(x) y \rangle, \tag{II.12}$$

for all $x, y \in \mathbb{R}^d$, and $j \in \mathbb{N}$. □

Lemma II.5. *Suppose that Assumption II.4 holds. Then for any $x, y \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p > 0$,*

$$m\|y\|^2 \leq \left\langle y, \nabla^2 \psi_p(x) y \right\rangle, \tag{II.13}$$

with m as in Assumption II.4.

Proof. From (II.10) and (II.12), we obtain that

$$\begin{aligned}
\left\langle y, \nabla^2 \psi_p(x) y \right\rangle &= \sum_{j \in Q} \mu_p^j(x) \left\langle y, \nabla^2 f^j(x) y \right\rangle + p \sum_{j \in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \nabla f^j(x)^T y \right\rangle \\
&- p \left\langle y, \left[\sum_{j \in Q} \mu_p^j(x) \nabla f^j(x) \right] \left[\sum_{j \in Q} \mu_p^j(x) \nabla f^j(x) \right]^T y \right\rangle \\
&= \sum_{j \in Q} \mu_p^j(x) \left\langle y, \nabla^2 f^j(x) y \right\rangle + p \sum_{j \in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \right\rangle^2 \\
&- p \left\langle y, \left[\sum_{j \in Q} \mu_p^j(x) \nabla f^j(x) \right] \right\rangle^2 \\
&\geq m \|y\|^2 + p \sum_{j \in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \right\rangle^2 - p \left\langle y, \left[\sum_{j \in Q} \mu_p^j(x) \nabla f^j(x) \right] \right\rangle^2 .
\end{aligned}$$

Hence, we only need to show that the difference of the last two terms is nonnegative. Let $g : \mathbb{R}^d \to \mathbb{R}$ be the convex function defined as $g(z) = \langle y, z \rangle^2$ for $y, z \in \mathbb{R}^d$. It follows from Jensen's inequality (see for example p. 6 of Urruty & Baptiste, 1996) that

$$\sum_{j \in Q} \mu_p^j(x) g\left(\nabla f^j(x) \right) \geq g \left(\sum_{j \in Q} \mu_p^j(x) \nabla f^j(x) \right). \tag{II.14}$$

Since $p > 0$, the result follows. □

For any matrix $A \in \mathbb{R}^{m \times n}$, we adopt the matrix norm $\|A\| \stackrel{\triangle}{=} \max_{\|u\|=1} \|Au\|$, where $u \in \mathbb{R}^n$. Under Assumption II.4(i), $|f^j(x)|$, $\|\nabla f^j(x)\|$, and $\|\nabla^2 f^j(x)\|$ are bounded on bounded subsets of $\mathbb{R}^d$ for given $j \in \mathbb{N}$.

Assumption II.6. *For any bounded set $S \subset \mathbb{R}^d$, there exists a $K \in (0, \infty)$ such that $\max\{|f^j(x)|, \|\nabla f^j(x)\|, \|\nabla^2 f^j(x)\|\} \leq K$ for all $x \in S, j \in \mathbb{N}$.* □

The assumption above holds for example under standard assumptions when $f^j(\cdot)$, $j \in \mathbb{N}$, arise from discretization of semi-infinite max functions. Under this assumption, we obtain the following useful result.

Lemma II.7. *Suppose that Assumptions II.4(i) and II.6 hold. Then for every bounded set $S \subset \mathbb{R}^d$,*

$$\left\langle y, \nabla^2 \psi_p(x) y \right\rangle \leq pL \|y\|^2 \tag{II.15}$$

for all $x \in S, y \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p \geq 1$, where $L = K + 2K^2$, with K as in Assumption II.6.

Proof. From the theory of matrix algebra, for matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times r}$, and vector $x \in \mathbb{R}^n$, we have that $\|Ax\| \leq \|A\|\|x\|$, $\|AB\| \leq \|A\|\|B\|$, and $\|xx^T\| = \|x\|^2$ (see for example p. 26 of Gill, Murray, & Wright, 1991). We consider each of the three terms of $\nabla^2\psi_p(\cdot)$; see (II.10). Recall that $\sum_{j\in Q} \mu_p^j(x) = 1$ for all $x \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p > 0$. For any $x \in S$, $y \in \mathbb{R}^d$, and $q \in \mathbb{N}$, under Assumption II.6, we obtain for the first term that

$$\begin{aligned} \left\langle y, \sum_{j\in Q} \mu_p^j(x)\nabla^2 f^j(x) y \right\rangle &\leq \|y\| \left\| \left(\sum_{j\in Q} \mu_p^j(x)\nabla^2 f^j(x) \right) y \right\| \\ &\leq \|y\|^2 \sum_{j\in Q} \mu_p^j(x) \left\|\nabla^2 f^j(x)\right\| \leq K\|y\|^2, \end{aligned} \tag{II.16}$$

where K is the constant in Assumption II.6 corresponding to S. Next, for the second term of $\nabla^2\psi_p(\cdot)$,

$$\begin{aligned} \left\langle y, \sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T y \right\rangle &\leq \|y\|^2 \left\| \sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T \right\| \\ &\leq \|y\|^2 \left(\sum_{j\in Q} \mu_p^j(x) \left\|\nabla f^j(x)\nabla f^j(x)^T\right\| \right) \\ &\leq K^2\|y\|^2. \end{aligned} \tag{II.17}$$

For the third term, we obtain that

$$\begin{aligned} &-\left\langle y, \left[\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\right] \left[\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\right]^T y \right\rangle \\ &\leq \|y\|^2 \left\| \left[\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\right] \left[\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\right]^T \right\| \leq K^2\|y\|^2. \end{aligned} \tag{II.18}$$

Hence, for all $x \in S$, $y \in \mathbb{R}^d$, $q \in \mathbb{N}$ and $p \geq 1$, $\langle y, \nabla^2\psi_p(x)y\rangle \leq (K + pK^2 + pK^2)\|y\|^2 \leq p(K + 2K^2)\|y\|^2$. □

Lemma II.7 enables us to quantify the rate of convergence of the Armijo Gradient Method for (FMX$_p$), as a function of $p \geq 1$, which we consider next.

Proposition II.8. *Suppose that Assumptions II.4 and II.6 hold. For any bounded set $S \subset \mathbb{R}^d$, there exists a $k \in (0,1)$ such that the rate of convergence of the Armijo Gradient Method to solve* (FMX_p)*, initialized by $x_0 \in S$, is linear with coefficient $1 - k/p$ for any $p \geq 1$ and $q \in \mathbb{N}$. That is, for all sequences $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ generated by the Armijo Gradient Method when applied to* (FMX_p)*, for any $p \geq 1$, $q \in \mathbb{N}$, and $x_0 \in S$, we have that*

$$\frac{\psi_p(x_{i+1}) - \psi_p^*}{\psi_p(x_i) - \psi_p^*} \leq 1 - \frac{k}{p} \quad \text{for all } i \in \mathbb{N}_0. \tag{II.19}$$

Proof. It follows by Lemma II.5 and Assumption II.6, and the fact that $x_0 \in S$, that there exists a bounded set $S' \subset \mathbb{R}^d$ such that all sequences generated by Armijo Gradient Method on (FMX_p), initialized by $x_0 \in S$, are contained in S' for all $p \geq 1$, $q \in \mathbb{N}$, $x_0 \in S$. Let m be as in Assumption II.4 and K be the constant in Assumption II.6 corresponding to S'. In view of Lemmas II.5 and II.7,

$$m\|y\|^2 \leq \left\langle y, \nabla^2 \psi_p(x) y \right\rangle \leq pL\|y\|^2, \tag{II.20}$$

for all $x \in S'$, $y \in \mathbb{R}$, $q \in \mathbb{N}$, and $p \geq 1$, where $L = K + 2K^2$. Hence, we deduce from Theorem 1.3.7 of Polak (1997) that the rate of convergence for Armijo Gradient Method to solve (FMX_p) is linear with coefficient $1 - 4m\beta\alpha(1-\alpha)/(pL) \in (0,1)$ for all $p \geq 1$, $q \in \mathbb{N}$, $x_0 \in S$, where $\alpha, \beta \in (0,1)$ are the Armijo line search parameters. Hence,

$$k = 4m\beta\alpha(1-\alpha)/L, \tag{II.21}$$

which is less than unity because $\alpha(1-\alpha) \in (0, 1/4]$ and $m \leq L$ in view of (II.20). □

2. Complexity

The results above enable us to identify the complexity of Algorithm II.1 under the following assumption on the computational work required for function and gradient evaluations. We let $t_0 \stackrel{\triangle}{=} \psi(x_0) - \psi^*$ for a given $x_0 \in \mathbb{R}^d$ and $q \in \mathbb{N}$.

Assumption II.9. *There exist constants $a, b \in (0, \infty)$ such that for any $d \in \mathbb{N}$, $j \in \mathbb{N}$, and $x \in \mathbb{R}^d$, the computational work to evaluate either $f^j(x)$ or $\nabla f^j(x)$ is no larger than ad^b.* □

Assumption II.9 holds for all problem instances considered in this chapter (see Appendix A) and appears reasonable for many practical situations. The following result can easily be modified to account for other assumption about work per function and gradient evaluation.

Theorem II.10. *Suppose that Assumptions II.4, II.6, and II.9 hold, and that Algorithm II.1 terminates after n iterations with $\psi(x_n) - \psi^* \leq t$. Then for any $d \in \mathbb{N}$ and bounded set $S \subset \mathbb{R}^d$, there exist constants $c, c', t' \in (0, \infty)$ such that the computational work until termination for Algorithm II.1 is no larger than*

$$c \frac{q \log q \log \frac{c'}{\delta t}}{(1-\delta)t}, \tag{II.22}$$

for all $q \in \mathbb{N}, q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, t']$.

Proof. Let $q \geq 2$ and $t \in (0, \log q]$, which ensures that $p^* = (\log q)/[(1-\delta)t] > 1$. Thus, Proposition II.8 applies and the number of iterations of the Armijo Gradient Method to generate $\{x_i\}_{i=0}^n$ such that $\psi_{p^*}(x_n) - \psi_{p^*}^* \leq \delta t$ is no larger than

$$\left\lceil \frac{\log \frac{\delta t}{t_0}}{\log(1 - \frac{k}{p^*})} \right\rceil, \tag{II.23}$$

where k is the constant in Proposition II.8 corresponding to S and $\lceil \cdot \rceil$ denotes the ceiling operator. In view of Proposition II.3, x_n also satisfies $\psi(x_n) - \psi^* \leq t$. Since the main computational work in each iteration for the Armijo Gradient Method is to determine $\nabla \psi_{p^*}(x_i)$, it follows by Assumption II.9 that there exist $a, b < \infty$ such that the computational work in each iteration of the Armijo Gradient Method when applied to (FMX_{p^*}) is no larger than aqd^b. Thus, the computational work in Algorithm II.1 to termination at x_n is no larger than (II.23) multiplied by aqd^b. Let f^{1*} denote the minimum value of $f^1(\cdot)$, which is finite according to Assumption II.4. Let K be the constant in Assumption II.6 corresponding to S. We then find that $t_0 = \psi(x_0) - \psi^* \leq K - f^{1*} \triangleq c'$, for any $x_0 \in S$ and $q \in \mathbb{N}$. It follows that the computational work in Algorithm II.1 to termination at x_n is no larger than

$$aqd^b \left\lceil \frac{\log \frac{\delta t}{c'}}{\log(1 - \frac{k}{p^*})} \right\rceil \tag{II.24}$$

for any $q \in \mathbb{N}$, $q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, \log q]$. Since $\log x \leq x - 1$ for $x \in (0,1]$, it follows by the choice of p^* that the computational work in Algorithm II.1 to termination at x_n is no larger than

$$aqd^b \left\lceil \frac{\log \frac{\delta t}{c'}}{\log\left(1 - \frac{k(1-\delta)t}{\log q}\right)} \right\rceil \leq aqd^b \left\lceil \frac{\log \frac{c'}{\delta t}}{\frac{k(1-\delta)t}{\log q}} \right\rceil, \tag{II.25}$$

for all $q \in \mathbb{N}, q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, \min\{\log q, c'\}]$.

There exists a $t' \in (0, \min\{\log q, c'\}]$ such that $\frac{\log q \log \frac{c'}{\delta t}}{k(1-\delta)t} \geq \frac{1}{2}$ for all $t \in (0, t']$, $q \in \mathbb{N}, q \geq 2$, and $\delta \in (0,1)$. This then implies that for all $q \in \mathbb{N}, q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, t']$,

$$aqd^b \left\lceil \frac{\log q \log \frac{c'}{\delta t}}{k(1-\delta)t} \right\rceil \leq 2aqd^b \left(\frac{\log q \log \frac{c'}{\delta t}}{k(1-\delta)t} \right) = \frac{2ad^b}{k} \left(\frac{q \log q \log \frac{c'}{\delta t}}{(1-\delta)t} \right). \tag{II.26}$$

Since k (see (II.21)) only depends on m from Assumption II.4, K from Assumption II.6, and user-defined parameters, the conclusion follows. □

We deduce from Theorem II.10 and its proof that the number of iterations of Algorithm II.1 required to achieve a solution with value within t of the optimal value of (FMX) is $O((1/t)\log 1/t)$ for fixed $q \geq 2$, $d \in \mathbb{N}$, and $\delta \in (0,1)$. This is worse than for example the Pshenichnyi-Pironneau-Polak (PPP) min-max algorithm (Algorithm 2.4.1 in Polak, 1997) and the modified conjugate gradient method on pp. 282-283 of Nemirovski and Yudin (1983), , which achieves $O(\log 1/t)$. The SQP algorithm in Zhou and Tits (1996) may also require a low number of iterations as it converges superlinearly, but its complexity in t is unknown. The larger number of iterations for Algorithm II.1 is caused by the fact that the Armijo Gradient Method exhibits slower rate of convergence as p increases (see Proposition II.8) and a larger p is required in Algorithm II.1 for a smaller t.

We next discuss the complexity of smoothing algorithms as compared to the SQP algorithms. We consider a sequence of finite minimax problems with the same number of variables d, but with an increasing number of functions q. This occurs

for example, in the solution of semi-infinite minimax problems using discretization algorithms, which we discuss in Chapter III.

When we also include the work per iteration of Algorithm II.1, we see from Theorem II.10 that for fixed $t \in (0, t']$, $d \in \mathbb{N}$, and $\delta \in (0, 1)$, the complexity is $O(q \log q)$. For comparison, the complexity of SQP and PPP algorithms to achieve a near-optimal solution of (FMX) is larger as we see next.

The main computational work in an iteration of an SQP algorithm involves solving a convex QP with $d + 1$ variables and q inequality constraints (Zhou & Tits, 1996). Introducing slack variables to convert into standard form, this subproblem becomes a convex QP with $d + 1 + q$ variables and q equality constraints. Based on Monteiro and Adler (1989), the computational work to solve the converted QP is $O((d + 1 + q)^3)$. Assuming that the number of iterations an SQP algorithm needs to achieve a near-optimal solution of (FMX) is $O(1)$, for fixed $t \in (0, t']$ and $d \in \mathbb{N}$, the complexity of an SQP algorithm to achieve a near-optimal solution of (FMX) is no better than $O(q^3)$. The same result holds for the PPP algorithm. This complexity, when compared with $O(q \log q)$ of Algorithm II.1, indicates that smoothing algorithms may be more efficient than SQP and PPP algorithms for (FMX) with large q. We carry out a comprehensive numerical comparison of smoothing algorithms with SQP and PPP algorithms in Section II.E. We note that the modified conjugate gradient method on pp. 282-283 of Nemirovski and Yudin (1983), may also have a low complexity in q, but this depends on its implementation and the method is only applicable to convex problems.

3. Optimal Parameter Choice

We see from Theorem II.10 that the computational work in Algorithm II.1 depends on the algorithm parameter δ. In this subsection, we find an "optimal" choice of δ. A direct minimization of (II.22) with respect to δ appears difficult and thus, we carry out a rate analysis and determine an optimal δ in that context.

The notation $t \downarrow 0$ means t approaches zero from above. We first consider the

situation as $t \downarrow 0$ and let $\delta_t \in (0,1)$ be a choice of δ in Algorithm II.1 for a specific t. For fixed $d \in \mathbb{N}$, $q \in \mathbb{N}$, $q \geq 2$, $S \subset \mathbb{R}^d$, and $x_0 \in S$, let c and c' be as in Theorem II.10 and let w_t denote (II.22) viewed as a function of $t > 0$, with δ replaced by δ_t, i.e.,

$$w_t \triangleq \tilde{c} \frac{\log \frac{c'}{\delta_t t}}{(1-\delta_t)t} \tag{II.27}$$

with $\tilde{c} = cq \log q$ for all $t > 0$. The next result shows that the choice of $\{\delta_t \in (0,1) \mid t > 0\}$ influences the rate with which $w_t \to \infty$, as $t \downarrow 0$. However, any constant δ_t for all $t > 0$ results in the slowest possible rate of increase in w_t, an asymptotic rate of $1/t$, as $t \downarrow 0$.

Theorem II.11. *For any* $\{\delta_t \in (0,1) \mid t > 0\}$,

$$\limsup_{t \downarrow 0} \frac{\log w_t}{\log t} \leq -1. \tag{II.28}$$

If $\delta_t = a \in (0,1)$ *for all* $t > 0$, *then*

$$\lim_{t \downarrow 0} \frac{\log w_t}{\log t} = -1. \tag{II.29}$$

Proof. There exists a $t_1 \in (0, \infty)$ such that $\log \frac{c'}{\delta_t t} \geq 1$ for all $t \in (0, t_1]$ and any $\{\delta_t \in (0,1) \mid t > 0\}$. Hence, for any $t \in (0, \min\{1, t_1\})$ and $\delta_t \in (0,1)$,

$$\begin{aligned} \frac{\log w_t}{\log t} &= \frac{\log \tilde{c}}{\log t} + \frac{\log \log \frac{c'}{\delta_t t}}{\log t} - \frac{\log(1-\delta_t)}{\log t} - \frac{\log t}{\log t} \\ &\leq \frac{\log \tilde{c}}{\log t} - 1, \end{aligned} \tag{II.30}$$

and the first part follows. Taking limits in (II.30), with $\delta_t = a$, yields the second part. □

We next consider the situation as $q \to \infty$ and, similar to above, let $\delta_q \in (0,1)$ be a choice of δ in Algorithm II.1 for a specific $q \in \mathbb{N}$. For fixed $d \in \mathbb{N}$ and $S \subset \mathbb{R}^d$, let c and c' be as in Theorem II.10. There exists a $t_1 \in (0, \infty)$ such that $\log(c/t) \geq 0$ and $\log(c'/t) \geq 1$ for all $t \in (0, t_1]$. For any given $q \in \mathbb{N}$, $q \geq 2$ and $t \in (0, t_1]$, let w_q denote (II.22) viewed as a function of q, with δ replaced by δ_q, i.e.,

$$w_q \triangleq \left(\frac{c}{t}\right) \frac{q \log q \log \frac{c'}{\delta_q t}}{(1-\delta_q)}. \tag{II.31}$$

The next result shows that the choice of $\{\delta_q\}_{q=2}^{\infty}$ influences the rate with which $w_q \to \infty$, as $q \to \infty$. However, for sufficiently small tolerance $t > 0$, as above, any constant choice of δ_q for all $q \in \mathbb{N}$ results in the slowest possible rate of increase in w_q, as $q \to \infty$. Hence, any constant $\delta \in (0,1)$ in Algorithm II.1 is optimal in this sense and results in the asymptotic rate of q, as $q \to \infty$.

Theorem II.12. *For any sequence of $\{\delta_q\}_{q=3}^{\infty}$, with $\delta_q \in (0,1)$, we have that*

$$\frac{\log w_q}{\log q} \geq 1 \tag{II.32}$$

for all $q \in \mathbb{N}$, $q \geq 3$, and $t \in (0, t_1]$. If $\delta_q = a$, where $a \in (0,1)$ is a constant, then

$$\lim_{q\to\infty} \frac{\log w_q}{\log q} = 1. \tag{II.33}$$

Proof. For $q \geq 3$,

$$\begin{aligned} \frac{\log w_q}{\log q} &= \frac{\log \frac{c}{t}}{\log q} + \frac{\log q}{\log q} + \frac{\log\log q}{\log q} + \frac{\log\log \frac{c'}{\delta_q t}}{\log q} - \frac{\log(1-\delta_q)}{\log q} \\ &\geq \frac{\log \frac{c}{t}}{\log q} + 1 + \frac{\log\log \frac{c'}{\delta_q t}}{\log q}. \end{aligned} \tag{II.34}$$

Since w_q is defined only for $t \in (0, t_1]$, and $\log(c/t) \geq 0$ and $\log(c'/t) \geq 1$ for all $t \in (0, t_1]$, it follows that $(\log w_q)/\log q \geq 1$ for all $q \geq 3$, $t \in (0, t_1]$, and $\{\delta_q\}_{q=3}^{\infty}$. The proof for the second part follows from taking the limit in (II.34). □

4. Rate of Convergence

The previous subsection considers the effect of the algorithm parameter δ on the computational work required in Algorithm II.1. This parameter defines the precision parameter through the relationship $p^* = (\log q)/((1-\delta)t)$; see Step 1 of Algorithm II.1. In this subsection, we do not restrict Algorithm II.1 to this class of choices for p^* and consider any positive value of the precision parameter. In particular, we examine the progress made by Algorithm II.1 after n iterations for different choices of p^*. Since the choice may depend on n, we denote by p_n the precision parameter used in Algorithm II.1 when terminated after n iterations. We examine the rate of

decay of an error bound on $\psi(x_n) - \psi^*$, and also determine the "optimal choice" of p_n that produces the fastest rate of decay of the error bound as $n \to \infty$.

Suppose that Assumptions II.4 and II.6 hold. For a given bounded set $S \subset \mathbb{R}^d$, let k be as in Proposition II.8 and let $\{x_i\}_{i=0}^n$, with $x_0 \in S$, be a sequence generated by Algorithm II.1 using $p^* = p_n$ for some $p_n > 0$. Then in view of (II.7) and Proposition II.8,

$$\begin{aligned} \psi(x_n) - \psi^* &\leq \psi_{p_n}(x_n) - \psi^*_{p_n} + \frac{\log q}{p_n} \\ &\leq \left(1 - \frac{k}{p_n}\right)^n \left(\psi_{p_n}(x_0) - \psi^*_{p_n}\right) + \frac{\log q}{p_n} \\ &\leq \left(1 - \frac{k}{p_n}\right)^n \left(\psi(x_0) - \psi^*\right) + \frac{2\log q}{p_n}. \end{aligned} \tag{II.35}$$

We want to determine the "best" $\{p_n\}_{n=1}^\infty$ such that the error bound on $\psi(x_n) - \psi^*$ defined by the right-hand side of (II.35) decays as fast as possible as $n \to \infty$. We denote that error bound by e_n, i.e., for any $n \in \mathbb{N}$,

$$e_n \triangleq t_0 \left(1 - \frac{k}{p_n}\right)^n + \frac{2\log q}{p_n}. \tag{II.36}$$

We need the following trivial technical result.

Lemma II.13. *For $x \in [0, 1/2]$, $-2x \leq \log(1-x) \leq -x$.* □

We next obtain that e_n asymptotically decays with a rate no faster than $1/n$, as $n \to \infty$, regardless of the choice of p_n, and that rate is attained with a particular choice of p_n.

Theorem II.14. *The following statements hold for e_n in (II.36):*

(i) For any $\{p_n\}_{n=1}^\infty$, with $p_n \geq 1$ for all $n \in \mathbb{N}$, $\liminf_{n\to\infty} \log e_n / \log n \geq -1$.

(ii) If $p_n = \zeta n / \log n$ for all $n \in \mathbb{N}$, with $\zeta \in (0, k]$, then $\lim_{n\to\infty} \log e_n / \log n = -1$.

(iii) If $p_n = n^{1-\nu} / \log n$ for all $n \in \mathbb{N}$, with $\nu \in (0, 1)$, then $\lim_{n\to\infty} \log e_n / \log n = -1 + \nu$.

Proof. For any $n \in \mathbb{N}$, we see from (II.36) that

$$\begin{aligned}\log e_n &= \log\left(\exp\left[\log t_0 + n\log\left(1-\frac{k}{p_n}\right)\right] + \frac{2\log q}{p_n}\right)\\ &\geq \log\left(\max\left\{\exp\left[\log t_0 + n\log\left(1-\frac{k}{p_n}\right)\right], \frac{2\log q}{p_n}\right\}\right)\\ &= \max\left\{\log\left(\exp\left[\log t_0 + n\log\left(1-\frac{k}{p_n}\right)\right]\right), \log\frac{2\log q}{p_n}\right\}.\end{aligned}$$

Hence, for any $n \in \mathbb{N}$, $n > 1$,

$$\frac{\log e_n}{\log n} \geq \max\left\{\frac{\log t_0}{\log n} + \frac{n\log\left(1-\frac{k}{p_n}\right)}{\log n}, -\frac{\log p_n}{\log n} + \frac{\log 2}{\log n} + \frac{\log\log q}{\log n}\right\}. \tag{II.37}$$

Let $\epsilon > 0$. Then there exists a $n_0 \in \mathbb{N}$ such that $(\log\log q)/\log n \geq -\epsilon$ for all $n \geq n_0$. If $(\log p_n)/\log n \leq 1$ and $n \geq \max\{2, n_0\}$, then

$$\frac{\log e_n}{\log n} \geq -\frac{\log p_n}{\log n} + \frac{\log 2}{\log n} + \frac{\log\log q}{\log n} \geq -\frac{\log p_n}{\log n} - \epsilon \geq -1 - \epsilon. \tag{II.38}$$

Alternatively, suppose that $(\log p_n)/\log n > 1$. Hence, $n/p_n < 1$, and if $n \geq 2k$, then $k/p_n \in (0, 1/2]$. Based on Lemma II.13 and (II.37),

$$\frac{\log e_n}{\log n} \geq \frac{\log t_0}{\log n} + \frac{n\log\left(1-\frac{k}{p_n}\right)}{\log n} \geq \frac{\log t_0}{\log n} + \frac{n\left(-\frac{2k}{p_n}\right)}{\log n} \geq \frac{\log t_0}{\log n} - \frac{2k}{\log n} \tag{II.39}$$

for all $n \geq 2k$ such that $(\log p_n)/\log n > 1$. Thus, there exists a $n_1 \geq \max\{n_0, 2k\}$ such that

$$\frac{\log t_0}{\log n} - \frac{2k}{\log n} \geq -1 - \epsilon \tag{II.40}$$

for all $n \geq n_1$. Hence, for all $n \geq n_1$, $(\log e_n)/\log n \geq -1 - \epsilon$. Since ϵ is chosen arbitrarily, the first part follows. Next, we prove the second part of the theorem. From (II.36), with $p_n = \zeta n/\log n$, where $\zeta \in (0, k]$,

$$\log e_n = \log\left(\exp\left[\log t_0 + n\log\left(1-\frac{k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right). \tag{II.41}$$

There exists a $n_2 \in \mathbb{N}$ such that $(k\log n)/\zeta n \in [0, 1/2]$ for all $n \geq n_2$. Thus, by Lemma II.13,

$$\begin{aligned}&\log\left(\exp\left[\log t_0 + n\left(-\frac{2k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right) \leq \log e_n\\ \leq\ &\log\left(\exp\left[\log t_0 + n\left(-\frac{k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right)\end{aligned} \tag{II.42}$$

for all $n \geq n_2$. We first consider the lower bound in (II.42),

$$\begin{aligned} & \log\left(\exp\left[\log t_0 + n\left(-\frac{2k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right) \\ = \; & \log\left(\frac{2\log q\log n}{\zeta n}\left[\frac{\exp\left(\log t_0 + \log n^{-2k/\zeta}\right)}{\frac{2\log q\log n}{\zeta n}} + 1\right]\right) \\ = \; & \log\left(\frac{2\log q\log n}{\zeta n}\right) + \log\left(\frac{t_0\zeta n^{1-\frac{2k}{\zeta}}}{2\log q\log n} + 1\right). \end{aligned} \tag{II.43}$$

Since $\zeta \in (0, k]$, and by continuity of the $\log(\cdot)$ function,

$$\lim_{n\to\infty} \log\left(\frac{t_0\zeta n^{1-\frac{2k}{\zeta}}}{2\log q\log n} + 1\right) = 0. \tag{II.44}$$

Continuing from (II.43), and using (II.44), we obtain that

$$\begin{aligned} & \lim_{n\to\infty} \frac{\log\left(\frac{2\log q\log n}{\zeta n}\right) + \log\left(\frac{t_0\zeta n^{1-\frac{2k}{\zeta}}}{2\log q\log n} + 1\right)}{\log n} \\ = \; & \lim_{n\to\infty} \frac{\log 2 + \log\log q + \log\log n - \log\zeta - \log n}{\log n} = -1. \end{aligned} \tag{II.45}$$

Similar arguments yield that the upper bound in (II.42) also tends to -1, as $n \to \infty$. Hence, the second conclusion follows. The third part of the theorem follows by similar arguments. □

We see from Theorem II.14 that the "best" choice of p_n is $p_n = \zeta n/\log n$, with $\zeta \in (0, k]$, and that choice results in an asymptotic rate of decay of error bound of $1/n$. The constant k may be unknown as it depends on m of Assumption II.4 and K of Assumption II.6; see (II.21). Consequently, $p_n = \zeta n/\log n$ may be difficult to implement. Theorem II.14 shows that the choice $p_n = n^{1-\nu}/\log n$ with a small $\nu \in (0, 1)$ is almost as good (it results in asymptotic rate $1/n^{1-\nu}$ instead of rate $1/n$) and is independent of k.

Roughly speaking, a rate of decay of error bound of no better than $1/n$ indicated by Theorem II.14 means that the required number of iterations to achieve an error tolerance t increases at least at rate $1/t$ as t approaches zero. In view of

Theorem II.11, the rate $1/t$ is attained with the precision parameter choice in Step 1 of Algorithm II.1. Hence, the choice in Step 1 of Algorithm II.1 for the precision parameter cannot be improved.

Theorems II.11 and II.14 indicate that Algorithm II.1 may only converge sublinearly. In contrast, Theorem II.10 shows that smoothing algorithms may still be capable of yielding competitive run times against other algorithms when q is large due to low computational work per iteration. For smoothing algorithms to be competitive in empirical test, however, we need to go beyond the basic Algorithm II.1 and develop more sophisticated, adaptive precision-adjustment schemes as discussed next.

D. SMOOTHING ALGORITHMS AND ADAPTIVE PRECISION ADJUSTMENT

The previous section shows that the choice of precision parameter influences the rate of convergence, since the degree of ill-conditioning in (FMX_p) depends on the precision parameter. This section presents two smoothing algorithms with novel precision-adjustment schemes for (FMX). The results in Polak et al. (2003) and our preliminary numerical tests strongly indicate that adaptive precision-adjustment schemes are superior to static and open-loop schemes in their ability to avoid ill-conditioning. Thus, we focus on adaptive precision-adjustment schemes in our smoothing algorithms.

The first algorithm, Algorithm II.2 follows Algorithm 3.2 in Polak et al. (2008), but uses a much simpler scheme for precision adjustment. The second algorithm, Algorithm II.3, adopts a novel line-search rule that aims to ensure descent in $\psi(\cdot)$ and, if that is not possible, increases the precision parameter. Previous smoothing algorithms (Polak et al., 2003, 2008) do not check for descent in $\psi(\cdot)$. The new algorithms implement active-set strategies adapted from Polak et al. (2008).

We use the following notation. The ϵ-active set, $\epsilon > 0$, is denoted by

$$Q_\epsilon(x) \stackrel{\triangle}{=} \{j \in Q | \psi(x) - f^j(x) \leq \epsilon\}. \tag{II.46}$$

As in Algorithm 3.2 of Polak et al. (2008), we compute a search direction using a $d \times d$ matrix $B_{p\Omega}(x)$. We consider two options. When

$$B_{p\Omega}(x) = I, \tag{II.47}$$

the $d \times d$ identity matrix, the search direction is equivalent to the steepest descent direction. When

$$B_{p\Omega}(x) = \eta_{p\Omega}(x)I + H_{p\Omega}(x), \tag{II.48}$$

the search direction is a Quasi-Newton direction, where

$$H_{p\Omega}(x) \stackrel{\triangle}{=} p\left(\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T - \left(\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\right)\left(\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\right)^T\right), \tag{II.49}$$

$$\eta_{p\Omega}(x) \stackrel{\triangle}{=} \max\{0, \varphi - e_{p\Omega}(x)\}, \tag{II.50}$$

$\varphi > 0$, and $e_{p\Omega}(x)$ is the smallest eigenvalue of $H_{p\Omega}(x)$. The quantity $\eta_{p\Omega}(x)$ ensures that $B_{p\Omega}(x)$ is positive definite. The Quasi-Newton direction given in (II.48)-(II.50) is adopted from Polak et al. (2008). Polak et al. (2008) observe that when $p \to \infty$, the first term in the Hessian function (II.10) becomes negligible, thus they ignore the first term.

We next present the two algorithms and proofs for their convergence.

1. Smoothing Algorithm Based on Optimality Function

We first consider the following smoothing algorithm, with a simple adaptive precision-adjustment scheme.

Algorithm II.2.

Data: $x_0 \in \mathbb{R}^d$.

Parameters and Auxiliary Functions: $\alpha, \beta \in (0,1), p_0 \geq 1, \omega = (10\log q)/p_0$, function $B_{p\Omega}(\cdot)$ as in (II.47) or (II.48), $\epsilon_0 > 0, \xi > 1, \varsigma > 1$, $\varphi \geq 1$.

Step 1. Set $i = 0, j = 0, \Omega_0 = Q_{\epsilon_0}(x_0)$.

Step 2. Compute the search direction $h_{p_i\Omega_i}(x_i)$ by solving

$$B_{p_i\Omega_i}(x_i)h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{II.51}$$

Step 3. Compute the stepsize $\lambda_i = \beta^{k_i}$, where k_i is the largest integer k such that

$$\psi_{p_i\Omega_i}(x_i + \beta^k h_{p_i\Omega_i}(x_i)) - \psi_{p_i\Omega_i}(x_i) \leq -\alpha\beta^k \|h_{p_i\Omega_i}(x_i)\|^2 \tag{II.52}$$

and

$$\psi_{p_i\Omega_i}(x_i + \beta^k h_{p_i\Omega_i}(x_i)) - \psi(x_i + \beta^k h_{p_i\Omega_i}(x_i)) \geq -\omega. \tag{II.53}$$

Step 4. Set

$$x_{i+1} = x_i + \beta^{k_i} h_{p_i\Omega_i}(x_i), \tag{II.54}$$

$$\Omega_{i+1} = \Omega_i \cup Q_{\epsilon_i}(x_{i+1}). \tag{II.55}$$

Step 5. Enter Subroutine II.1, and go to Step 2 on exit from Subroutine II.1.

Subroutine II.1. Adaptive Precision-Parameter Adjustment using Optimality Function

If

$$\theta_{p_i\Omega_i}(x_{i+1}) \geq -\epsilon_i, \tag{II.56}$$

set $x_j^* = x_{i+1}$, set $p_{i+1} = \xi p_i$, set $\epsilon_{i+1} = \epsilon_i/\varsigma$, replace i by $i + 1$, replace j by $j + 1$, and exit Subroutine II.1.

Else, set $p_{i+1} = p_i$, set $\epsilon_{i+1} = \epsilon_i$, replace i by $i + 1$, and exit Subroutine II.1. □

Steps 1 to 4 of Algorithm II.2 are adopted from Algorithm 3.2 of Polak et al. (2008). We note the unusual choice of the right-hand side in (II.52), where $-\|h_{p_i\Omega_i}(x_i)\|^2$ is used instead of the conventional $\langle\nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle$. Test runs show that Algorithm II.2 with $-\|h_{p_i\Omega_i}(x_i)\|^2$ is slightly more efficient than with the conventional $\langle\nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle$. To allow direct comparison with Algorithm 3.2 of Polak et al. (2008), we use $-\|h_{p_i\Omega_i}(x_i)\|^2$ in Algorithm II.2.

The test in (II.53) prevents the construction of a point x_{i+1} where $\psi(x_{i+1})$ is much greater than $\psi(x_i)$ during the early iterations when the set Ω_i is small; see Polak et al. (2008).

The key difference between Algorithm II.2 and Algorithm 3.2 of Polak et al. (2008) is the simplified scheme to adjust p_i in Subroutine II.1. This difference calls for a different proof of convergence as compared to Polak et al. (2008), and will be based on consistent approximation; see Section I.D.3. Let $\mathcal{P}$ denote an increasing sequence of positive real numbers that approach infinity.

The following result shows that the pairs $((\mathrm{FMX}_{p\Omega}), \theta_{p\Omega}(\cdot))$ in the sequence $\{((\mathrm{FMX}_{p\Omega}), \theta_{p\Omega}(\cdot))\}_{p\in\mathcal{P}}$ are indeed consistent approximations to $((\mathrm{FMX}_{\Omega}), \theta_{\Omega}(\cdot))$. This is subsequently used in the proof of convergence of Algorithm II.2.

Theorem II.15. *Suppose that Assumption II.4(i) holds. Then for any* $\Omega \subset \mathbb{N}$, *the pairs* $((\mathrm{FMX}_{p\Omega}), \theta_{p\Omega}(\cdot))$ *in the sequence* $\{((\mathrm{FMMP}_{p\Omega}), \theta_{p\Omega}(\cdot))\}_{p\in\mathcal{P}}$ *are consistent approximations to* $((\mathrm{FMX}_{\Omega}), \theta_{\Omega}(\cdot))$.

Proof. We follow the proofs of Lemmas 4.3 and 4.4 in Polak (2003), but simplify the arguments as Polak (2003) deals with min-max-min problems. According to Theorem 3.3.2 of Polak (1997), $(\mathrm{FMX}_{p_i\Omega})$ epi-converges to (FMX_{Ω}), as $i \to \infty$ if and only if (i) for any $x^* \in \mathbb{R}^d$, there exists a sequence $\{x_i\}_{i=0}^{\infty}$, with $x_i \in \mathbb{R}^d$, such that $x_i \to x^*$ and $p_i \to \infty$, as $i \to \infty$, $\limsup_{i\to\infty} \psi_{p_i\Omega}(x_i) \leq \psi_{\Omega}(x^*)$ and (ii) for any sequence $\{x_i\}_{i=0}^{\infty}$, such that $x_i \in \mathbb{R}^d$, $x_i \to x^* \in \mathbb{R}^d$, and $p_i \to \infty$ as $i \to \infty$, $\liminf_{i\to\infty} \psi_{p_i\Omega}(x_i) \geq \psi_{\Omega}(x^*)$.

(i) Let $x^* \in \mathbb{R}^d$. Construct a sequence $\{x_i\}_{i=0}^{\infty}$, where $x_i = x^*$ for all i. Obviously, $x_i \to x^*$ as $i \to \infty$. According to Proposition II.1(ii), $\psi_{p\Omega}(x) \to \psi_{\Omega}(x)$, as $p \to \infty$, this implies that $\limsup_{p\to\infty} \psi_{p\Omega}(x^*) = \liminf_{p\to\infty} \psi_{p\Omega}(x^*) = \psi_{\Omega}(x^*)$. Therefore, $\limsup_{i\to\infty} \psi_{p_i\Omega}(x_i) = \limsup_{i\to\infty} \psi_{p_i\Omega}(x^*) = \psi_{\Omega}(x^*)$.

(ii) Let $\{x_i\}_{i=0}^{\infty}$ and $\{p_i\}_{i=0}^{\infty}$ be arbitrary sequences such that $x_i \to x^*$, $x^* \in \mathbb{R}^d$, and $p_i \to \infty$, as $i \to \infty$. For any $t > 0$, there exists by continuity of $\psi_{\Omega}(\cdot)$ an i_0 such that $\psi_{\Omega}(x_i) - \psi_{\Omega}(x^*) < \frac{t}{2}$ for all $i \geq i_0$. Moreover, from (II.7), there

exists an i_1 such that $\psi_{p_i\Omega}(x) - \psi_\Omega(x) \leq \frac{\log(|\Omega|)}{p_i} < \frac{t}{2}$ for all $i \geq i_1$. Then for all $i \geq \max(i_0, i_1), \psi_{p_i\Omega}(x_i) - \psi_\Omega(x^*) = \psi_{p_i\Omega}(x_i) - \psi_\Omega(x_i) + \psi_\Omega(x_i) - \psi_\Omega(x^*) < t$. Hence, $\psi_{p_i\Omega}(x_i) \to \psi_\Omega(x^*)$.

We next consider the optimality functions. Let $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^\infty, p_i > 0$ for all i, be arbitrary sequences and $x^* \in \mathbb{R}^d$ be such that $x_i \to x^*$ and $p_i \to \infty$, as $i \to \infty$. Since $\mu_p^j(x) \in (0,1)$ for any $j \in \Omega$, $p > 0$, and $x \in \mathbb{R}^d$, $\{\mu_{p_i}(x_i)\}_{i=0}^\infty$ is a bounded sequence in $\mathbb{R}^{|\Omega|}$, and, according to the Bolzano-Weierstrass Theorem, there exists at least one convergent subsequence. For every such subsequence $K \subset \mathbb{N}_0$, there exists a $\mu_\infty \in \Sigma_\Omega$ such that $\mu_{p_i}(x_i) \to^K \mu_\infty$, as $i \to \infty$. Moreover, since $\mu_\infty \in \Sigma_\Omega$, $\sum_{j\in\Omega} \mu_\infty^j = 1$.

If $j \notin \hat{\Omega}(x^*)$, then there exist a $t > 0$ and $i_0 \in \mathbb{N}$ such that $f^j(x_i) - \psi_\Omega(x_i) \leq -t$ for all $i \geq i_0$. Hence, from (II.9), $\mu_{p_i}^j(x_i) \to 0$, as $i \to \infty$, and therefore $\mu_\infty^j = 0$. By continuity of $\nabla f^j(\cdot)$, $j \in \Omega$,

$$\theta_{p_i\Omega}(x_i) \to^K -\frac{1}{2}\|\sum_{j\in\Omega} \mu_\infty^j \nabla f^j(x^*)\|^2 \triangleq \theta_{\infty\Omega}(x^*), \tag{II.57}$$

as $i \to \infty$. Since $\mu_\infty \in \Sigma_\Omega$ and $\mu_\infty^j = 0$ for all $j \notin \hat{\Omega}(x^*)$, we find in view of (II.11) that

$$\theta_{\infty\Omega}(x^*) = -\sum_{j\in\Omega} \mu_\infty^j(\psi_\Omega(x^*) - f^j(x^*)) - \frac{1}{2}\|\sum_{j\in\Omega} \mu_\infty^j \nabla f^j(x^*)\|^2 \leq \theta_\Omega(x^*). \tag{II.58}$$

This completes the proof. □

The next result is identical to Lemma 3.1 in Polak et al. (2008).

Lemma II.16. *Suppose that $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ is a sequence constructed by Algorithm II.2. Then there exists an $i^* \in \mathbb{N}_0$ and a set $\Omega^* \subseteq Q$ such that the working sets Ω_i satisfy $\Omega_i = \Omega^*$ for all $i \geq i^*$.*

Proof. By construction, $\Omega_i \subseteq \Omega_{i+1}$ for all $i \in \mathbb{N}_0$. Since the set Q is finite, the lemma must be true. □

The following result ensures convergence of Algorithm II.2.

Theorem II.17. *Suppose that Assumption II.4(i) holds. Then any accumulation point $x^* \in \mathbb{R}^d$ of a sequence $\{x_j^*\}_{j=0}^{\infty} \subset \mathbb{R}^d$ constructed by Algorithm II.2 satisfies the first-order optimality condition $\theta(x^*) = 0$.*

Proof. Let $\Omega^* \subseteq Q$ and $i^* \in \mathbb{N}_0$ be as in Lemma II.16, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. As Algorithm II.2 has the form of Master Algorithm Model 3.3.12 in Polak (1997) for all $i \geq i^*$, we conclude based on Theorem 3.3.13 of Polak (1997) that any accumulation point x^* of a sequence $\{x_j^*\}_{j=0}^{\infty}$ constructed by Algorithm II.2 satisfies $\theta_{\Omega^*}(x^*) = 0$. The assumptions required to invoke Theorem 3.3.13 in Polak (1997):

(i) Continuity of $\psi_{\Omega^*}(\cdot), \psi_{p\Omega^*}(\cdot), \theta_{\Omega^*}(\cdot)$, and $\theta_{p\Omega^*}(\cdot)$, $p > 0$, which follows by Assumption II.4(i), Proposition II.1(i), Theorem 2.1.6 of Polak (1997), and Proposition II.1(iii), respectively.

(ii) The pairs $((\mathrm{FMX}_{p\Omega^*}), \theta_{p\Omega^*}(\cdot))$ in the sequence $\{((\mathrm{FMX}_{p\Omega^*}), \theta_{p\Omega^*}(\cdot))\}_{p\in\mathcal{P}}$ are consistent approximations to $((\mathrm{FMX}_{\Omega^*}), \theta_{\Omega^*}(\cdot))$, which follows by Theorem II.15.

(iii) If Steps 1 to 4 of Algorithm II.2 are applied repeatedly to $(\mathrm{FMX}_{p\Omega^*})$ with a fixed $p > 0$, then every accumulation point $\hat{x}$ of a sequence $\{x_k\}_{k=0}^{\infty}$ constructed must be a stationary point of $(\mathrm{FMX}_{p\Omega^*})$, i.e., $\theta_{p\Omega^*}(\hat{x}) = 0$, which follows by Theorem 3.2 in Polak et al. (2008).

Since $\theta_{\Omega^*}(x^*) = 0$, from (II.11), there exists a $\mu \in \Sigma_{\Omega^*}$ such that

$$\sum_{j\in\Omega^*} \mu^j(\psi_{\Omega^*}(x^*) - f^j(x^*)) + \tfrac{1}{2}\left\|\sum_{j\in\Omega^*} \mu^j \nabla f^j(x^*)\right\|^2 = 0. \tag{II.59}$$

Let $\pi \in \Sigma_Q$, $\pi^j = 0$ for $j \in Q - \Omega^*$, and $\pi^j = \mu^j$ for $j \in \Omega^*$. Thus, it follows from (II.11) that

$$\theta(x^*) \geq -\sum_{j\in Q} \pi^j(\psi(x^*) - f^j(x^*)) - \tfrac{1}{2}\left\|\sum_{j\in Q} \pi^j \nabla f^j(x^*)\right\|^2 = 0. \tag{II.60}$$

Since $\theta(\cdot)$ is a nonpositive function, the result follows. □

2. Smoothing Algorithm Using Cost Descent

Next, we consider the second smoothing algorithm, which determines the step-size based on the actual function $\psi(\cdot)$ rather than the smoothed function $\psi_{p\Omega}(\cdot)$.

Algorithm II.3.

Data: $x_0 \in \mathbb{R}^d$.

Parameters and Auxiliary Functions: $\alpha, \beta \in (0,1)$, function $B_{p\Omega}(\cdot)$ as in (II.47) or (II.48), $\epsilon > 0, \varphi \geq 1, p_0 \geq 1, \hat{p} \gg p_0, \kappa \gg 1, \xi > 1, \gamma > 0, \nu \in (0,1), \Delta p \geq 1$.

Step 0. Set $i = 0, \Omega_0 = Q_\epsilon(x_0), k_{-1} = 0$.

Step 1. Compute $B_{p_i\Omega_i}(x_i)$ and its largest eigenvalue $\sigma^{\max}_{p_i\Omega_i}(x_i)$. If

$$\sigma^{\max}_{p_i\Omega_i}(x_i) \geq \kappa, \tag{II.61}$$

compute the search direction

$$h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{II.62}$$

Else, compute the search direction $h_{p_i\Omega_i}(x_i)$ by solving the equation

$$B_{p_i\Omega_i}(x_i)h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{II.63}$$

Step 2a. Compute a tentative Armijo stepsize based on working set Ω_i, starting from the eventual stepsize of the previous iterate k_{i-1}, i.e., determine

$$\begin{aligned} &\lambda_{p_i\Omega_i}(x_i) = \max_{l\in\{k_{i-1},k_{i-1}+1,\ldots\}} \beta^l \\ &\text{s.t. } \psi_{p_i\Omega_i}(x_i + \beta^l h_{p_i\Omega_i}(x_i)) - \psi_{p_i\Omega_i}(x_i) \leq \alpha\beta^l\langle\nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle. \end{aligned} \tag{II.64}$$

Set

$$y_i = x_i + \beta^l h_{p_i\Omega_i}(x_i). \tag{II.65}$$

Step 2b. Forward track from y_i along direction $h_{p_i\Omega_i}(x_i)$ as long as $\psi(\cdot)$ continues to decrease using the following subroutine.

Substep 0. Set $l' = l$,

$$z_{il'} = x_i + \beta^{l'} h_{p_i\Omega_i}(x_i) \text{ and } z_{il'-1} = x_i + \beta^{l'-1} h_{p_i\Omega_i}(x_i). \tag{II.66}$$

Substep 1. If

$$\psi(z_{il'-1}) < \psi(z_{il'}), \tag{II.67}$$

replace l' by $l'-1$, set $z_{il'-1} = x_i + \beta^{l'-1} h_{p_i\Omega_i}(x_i)$, and repeat Substep 1.

Else, set $z_i = z_{il'}$.

Substep 2. If $p_i \leq \hat{p}$, go to Step 3. Else, go to Step 4.

Step 3. If

$$\psi(z_i) - \psi(x_i) \leq -\frac{\gamma}{{p_i}^{\nu}}, \tag{II.68}$$

set $x_{i+1} = z_i, p_{i+1} = p_i, k_i = l'$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1.

Else, replace p_i by ξp_i, replace Ω_i by $\Omega_i \cup Q_\epsilon(z_i)$, and go to Step 1.

Step 4. If (II.68) holds, set $x_{i+1} = z_i, k_i = l'$, set $p_{i+1} = p_i + \Delta p$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1.

Else, set $x_{i+1} = y_i, k_i = l$, set $p_{i+1} = p_i + \Delta p$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1. □

As is standard in stabilized Newton methods (see for example Section 1.4.4 of Polak, 1997), Algorithm II.3 switches to the steepest descent direction if $B_{p\Omega}(\cdot)$ is given by (II.48) and the largest eigenvalue of $B_{p\Omega}(\cdot)$ is large; see Step 1. Compared to Algorithm 3.2 in Polak et al. (2008), which increases p when $\|\nabla\psi_{p_i\Omega_i}(x_i)\|$ is small, Algorithm II.3 increases the precision parameter only when it does not produce sufficient descent in $\psi(\cdot)$, as verified by the test (II.68) in Steps 3 and 4 of Algorithm II.3. A small precision parameter may produce an ascent direction in $\psi(\cdot)$ due to the poor accuracy of $\psi_{p_i\Omega_i}(\cdot)$. Thus, insufficient descent is a signal that the precision parameter may be too small. All existing smoothing algorithms only ensure that $\psi_{p_i\Omega_i}(\cdot)$ decreases at each iteration, but do not ensure descent in $\psi(\cdot)$. Another change compared to Polak et al. (2003, 2008) relates to the line search. All smoothing algorithms are susceptible to ill-conditioning and small stepsizes. To counteract this difficulty, Algorithm II.3 moves forward along the search direction starting from the Armijo step, and stops when the next step is not a descent step in $\psi(\cdot)$; see Step 2b.

Algorithm II.3 has two rules for increasing p_i. In the early stages of the calculations, i.e., when $p_i \leq \hat{p}$, if sufficient descent in $\psi(\cdot)$ is achieved when moving

from x_i to z_i ((II.68) satisfied), then Algorithm II.3 sets the next iterate x_{i+1} to z_i, retain the current value of the precision parameter as progress is made towards the optimal solution of (FMX). However, if (II.68) fails, then there is insufficient descent and the precision parameter or the working set needs to be modified to generate a better search direction in the next iteration. In late stages of the calculations, i.e., $p_i > \hat{p}$, Algorithm II.3 accepts every new point generated, even those with insufficient descent, and increases the precision parameter with a constant value.

The next lemma is similar to Lemma II.16.

Lemma II.18. *Suppose that $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ is a sequence constructed by Algorithm II.3. Then there exists an $i^* \in \mathbb{N}_0$ and a set $\Omega^* \subseteq Q$ such that the working sets Ω_i satisfy $\Omega_i = \Omega^*$ and $\psi_{\Omega^*}(x_i) = \psi(x_i)$ for all $i \geq i^*$.*

Proof. The first part of the proof follows exactly from the proof for Lemma II.16. Next, since $\widehat{Q}(x_i) \subseteq \Omega_i$ for all i; see Steps 3 and 4 of Algorithm II.3, $\psi_{\Omega^*}(x_i) = \psi(x_i)$ for all $i \geq i^*$. □

Lemma II.19. *Suppose that Assumption II.4(i) holds, and that the sequences $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^{\infty} \subset \mathbb{R}$ are generated by Algorithm II.3. Then the following properties hold: (i) the sequence $\{p_i\}_{i=0}^{\infty}$ is monotonically increasing; (ii) if the sequence $\{x_i\}_{i=0}^{\infty}$ has an accumulation point, then $p_i \to \infty$ as $i \to \infty$, and $\sum_{i=0}^{\infty} 1/p_i = +\infty$.*

Proof. We follow the framework of the proof for Lemma 3.1 of Polak et al. (2003). (i) The precision parameter is adjusted in Steps 3 and 4 of Algorithm II.3. In Step 3, if (II.68) is satisfied, then $p_{i+1} = p_i$; if (II.68) fails, p_i is replaced by $\xi p_i > p_i$. In Step 4, $p_{i+1} = p_i + \Delta p \geq p_i + 1 > p_i$.

(ii) Suppose that Algorithm II.3 generates the sequence $\{x_i\}_{i=0}^{\infty}$ with accumulation point $x^* \in \mathbb{R}^d$, but $\{p_i\}_{i=0}^{\infty}$ is bounded from above. The existence of an upper bound on p_i implies that $p_i \leq \hat{p}$ for all $i \in \mathbb{N}_0$, because if not, Algorithm II.3 will enter Step 4 the first time at some iteration $i' \in \mathbb{N}_0$, and re-enter Step 4 for all $i > i'$,

and $p_i \to \infty$ as $i \to \infty$. Thus, the existence of an upper bound on p_i implies that Algorithm II.3 must never enter Step 4.

The existence of an upper bound on p_i also implies that there exists an iteration $i^* \in \mathbb{N}_0$ such that (II.68) is satisfied for all $i > i^*$, because if not, p_i will be replaced by ξp_i repeatedly, and $p_i \to \infty$ as $i \to \infty$. This means that $\psi(x_{i+1}) - \psi(x_i) \leq -\gamma/{p_i}^\nu$ for all $i > i^*$. Since $p_i \leq \hat{p}$ for all $i \in \mathbb{N}_0$, $\psi(x_i) \to -\infty$ as $i \to \infty$. However, by continuity of $\psi(\cdot)$, and x^* being an accumulation point, $\psi(x_i) \to^K \psi(x^*)$, where $K \subset \mathbb{N}_0$ is some infinite subset. This is a contradiction, so $p_i \to \infty$.

Next, we prove that $\sum_{i=0}^{\infty} 1/p_i = +\infty$. Since $p_i \to \infty$, there exist an iteration $i^* \in \mathbb{N}_0$ such that $p_i > \hat{p}$ for all $i \geq i^*$. This means that the precision parameter is adjusted by the rule $p_{i+1} = p_i + \Delta p$ for all $i \geq i^*$. The proof is complete by the fact that $\sum_{i=1}^{\infty} 1/i = \infty$. □

Lemma II.20. *Suppose that Assumption II.4(i) holds. Then for every bounded set $S \subset \mathbb{R}^d$ and parameters $\alpha, \beta \in (0,1)$, there exist a $K < \infty$ such that, for all $p \geq 1$, $\Omega \subseteq Q$, and $x \in S$,*

$$\psi_{p\Omega}(x + \lambda_{p\Omega}(x) h_{p\Omega}(x)) - \psi_{p\Omega}(x) \leq \frac{-\alpha K \|\nabla \psi_{p\Omega}(x)\|^2}{p}, \tag{II.69}$$

where $\lambda_{p\Omega}(x)$ is the stepsize defined by (II.64) and $h_{p\Omega}(x)$ is the search direction as defined by (II.62) or (II.63), with p_i replaced by p, Ω_i replaced by Ω, and x_i replaced by x.

Proof. If $h_{p\Omega}(x)$ is given by (II.63) with $B_{p\Omega}(x)$ as in (II.47), then the result follows by the same arguments as in the proof for Lemma 3.2 of Polak et al. (2003). If $h_{p\Omega}(x)$ is given by (II.63) with $B_{p\Omega}(x)$ as in (II.48), then the result follows by similar arguments as in the proof for Lemma 3.4 of Polak et al. (2003), but the argument deviates to account for (i) the lower bound on the eigenvalues of $B_{p\Omega}(x)$ takes on the specific value of 1 in Algorithm II.3, and (ii) we consider an arbitrary $\Omega \subseteq Q$.

Based on Assumption II.4(i), (II.8), and the assumption that S is a bounded set, there exists a constant $M < \infty$ such that $\|\nabla \psi_{p\Omega}(x)\| \leq M$, for all $p \geq 1$, $\Omega \subseteq Q$,

and $x \in S$. Let

$$S_B \stackrel{\triangle}{=} \left\{ x \in \mathbb{R}^d | \|x - x'\| \leq M, x' \in S \right\}, \tag{II.70}$$

and $L < \infty$ be the constant corresponding to S_B such that (II.15) holds for all $x \in S_B, y \in \mathbb{R}^d$ and $p \geq 1$. For any real $d \times d$ matrix A, let $\|A\|$ denote its induced matrix norm as defined on p. 20. If A is symmetric,

$$\|A\| = \sigma^{\max}, \tag{II.71}$$

whenever $\sigma^{\max} \geq 0$, where $\sigma^{\max}$ is the largest eigenvalue of A; see for example p. 3 of Lang (2000). Now, suppose that $p \geq 1$, $\Omega \subseteq Q$, and $x \in S$ are such that $h_{p\Omega}(x) = -B_{p\Omega}(x)^{-1} \nabla \psi_{p\Omega}(x)$. Since all induced norms are consistent by definition, $\|h_{p\Omega}(x)\| \leq \|B_{p\Omega}(x)^{-1}\| \|\nabla \psi_{p\Omega}(x)\|$.

By construction, $B_{p\Omega}(x)$ is symmetric and positive definite as the minimum eigenvalue of $B_{p\Omega}(x)$ is 1, because $\varphi \geq 1$, and based on (II.50). Thus, $B_{p\Omega}(x)^{-1}$ is symmetric and positive definite; see for example Bertsekas, Nedic, and Ozdaglar (2003, p. 16). Hence, using the fact that the eigenvalues of an inverse matrix are the reciprocals of the eigenvalues of the original matrix, and (II.71),

$$\|h_{p\Omega}(x)\| \leq \|B_{p\Omega}(x)^{-1}\| \|\nabla \psi_{p\Omega}(x)\| = \frac{\|\nabla \psi_{p\Omega}(x)\|}{\sigma^{\min}_{p\Omega}(x)}, \tag{II.72}$$

and

$$\langle \nabla \psi_{p\Omega}(x), h_{p\Omega}(x) \rangle \leq -\frac{\|\nabla \psi_{p\Omega}(x)\|^2}{\sigma^{\max}_{p\Omega}(x)}, \tag{II.73}$$

where $\sigma^{\min}_{p\Omega}(x)$ and $\sigma^{\max}_{p\Omega}(x)$ are the smallest and largest eigenvalues of $B_{p\Omega}(x)$ respectively. From Step 1 of Algorithm II.3, we see that the direction in (II.63) is selected only when $\sigma^{\max}_{p\Omega}(x) < \kappa$, and by construction according to (II.50), $\sigma^{\min}_{p\Omega}(x) \geq 1$. Hence, from (II.72) and (II.73),

$$\|h_{p\Omega}(x)\| \leq \|\nabla \psi_{p\Omega}(x)\| \tag{II.74}$$

and

$$\langle \nabla \psi_{p\Omega}(x), h_{p\Omega}(x) \rangle \leq -\frac{\|\nabla \psi_{p\Omega}(x)\|^2}{\kappa}. \tag{II.75}$$

It follows directly from (II.62) that (II.74) and (II.75) also hold when $h_{p\Omega}(x) = -\nabla\psi_{p\Omega}(x)$.

Next, for all $\lambda \in (0,1], x \in S$ and $p \geq 1$, using the Mean-Value Theorem (see for example Section 5.1.28 of Polak, 1997) and Lemma II.7, we have for some $s \in [0,1]$,

$$\begin{aligned}
&\psi_{p\Omega}(x + \lambda h_{p\Omega}(x)) - \psi_{p\Omega}(x) - \alpha\lambda\langle\nabla\psi_{p\Omega}(x), h_{p\Omega}(x)\rangle \\
&= \lambda(1-\alpha)\langle\nabla\psi_{p\Omega}(x), h_{p\Omega}(x)\rangle + \tfrac{1}{2}\lambda^2 \left\langle h_{p\Omega}(x), \nabla^2\psi_{p\Omega}\left(x + s\lambda h_{p\Omega}(x)\right) h_{p\Omega}(x)\right\rangle \\
&\leq \lambda(1-\alpha)\langle\nabla\psi_{p\Omega}(x), h_{p\Omega}(x)\rangle + \tfrac{1}{2}\lambda^2 pL\|h_{p\Omega}(x)\|^2 \\
&\leq -\lambda\|\nabla\psi_{p\Omega}(x)\|^2 \left[\frac{1-\alpha}{\kappa} - \tfrac{1}{2}\lambda pL\right]. \qquad \text{(II.76)}
\end{aligned}$$

Let

$$\lambda^* \triangleq \min\left\{1, \frac{2(1-\alpha)}{pL\kappa}\right\}. \qquad \text{(II.77)}$$

Then it follows from (II.77) that, for every $\lambda \in (0, \lambda^*]$, we have

$$\psi_{p\Omega}(x + \lambda h_{p\Omega}(x)) - \psi_{p\Omega}(x) - \alpha\lambda\langle\nabla\psi_{p\Omega}(x), h_{p\Omega}(x)\rangle \leq 0. \qquad \text{(II.78)}$$

Hence, by (II.78) and the stepsize rule in (II.64),

$$\lambda_{p\Omega}(x) \geq \beta\lambda^* \qquad \text{(II.79)}$$

for all $p \geq 1$, $\Omega \subseteq Q$, and $x \in S$. Consequently, by (II.64) and (II.79), we have that

$$\begin{aligned}
&\psi_{p\Omega}(x + \lambda_{p\Omega}(x)h_{p\Omega}(x)) - \psi_{p\Omega}(x) \\
&\leq -\alpha\lambda_{p\Omega}(x)\langle\nabla\psi_{p\Omega}(x), h_{p\Omega}(x)\rangle \\
&\leq -\alpha\min\left\{\beta, \frac{2\beta(1-\alpha)}{pL\kappa}\right\}\frac{\|\nabla\psi_{p\Omega}(x)\|^2}{\kappa} \qquad \text{(II.80)}
\end{aligned}$$

for all $p \geq 1$, $\Omega \subseteq Q$, and $x \in S$. Hence, the conclusion follows with

$$K = \min\left\{\beta, \frac{2\beta(1-\alpha)}{L\kappa}\right\}. \qquad \text{(II.81)}$$

This completes the proof. □

Lemma II.21. *Suppose that Assumption II.4(i) holds and that $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ is a bounded sequence generated by Algorithm II.3. Let $\Omega^* \subseteq Q$ and $i^* \in \mathbb{N}_0$ be as in Lemma II.18, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. Then there exists an accumulation point $x^* \in \mathbb{R}^d$ of the sequence $\{x_i\}_{i=0}^{\infty}$ such that $\theta_{\Omega^*}(x^*) = 0$.*

Proof. For the sake of contradiction, we assume that there exist a $\rho > 0$ such that

$$\liminf_{i\to\infty} \|\nabla\psi_{p_i\Omega^*}(x_i)\| \geq \rho. \tag{II.82}$$

Since $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence, it has at least one accumulation point according to the Bolzano-Weierstrass Theorem. Hence, by Lemma II.19, $p_i \to \infty$, as $i \to \infty$. Consider two cases, $x_{i+1} = y_i$ or $x_{i+1} = z_i$ in Algorithm II.3.

If $x_{i+1} = y_i$, by Lemma II.20, there exists an $M < \infty$ such that

$$\psi_{p_i\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \tag{II.83}$$

for $i \geq i^*$. Hence,

$$\begin{aligned}\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) &= \psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_{i+1}) + \psi_{p_i\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i)\\ &\leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i},\end{aligned} \tag{II.84}$$

for $i \geq i^*$, where we have used the fact from Proposition II.1(i) that

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) \leq \psi_{p_i\Omega^*}(x_{i+1}), \tag{II.85}$$

for $i \geq i^*$, because $p_{i+1} \geq p_i$ from Lemma II.19.

Next, if $x_{i+1} = z_i$, then (II.68) is satisfied. It follows from (II.7) and Lemma II.18 that,

$$\begin{aligned}\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) &\leq \psi_{\Omega^*}(x_{i+1}) + \frac{\log|\Omega^*|}{p_{i+1}} - \psi_{\Omega^*}(x_i)\\ &= \psi(x_{i+1}) + \frac{\log|\Omega^*|}{p_{i+1}} - \psi(x_i)\\ &\leq -\frac{\gamma}{{p_i}^{\nu}} + \frac{\log|\Omega^*|}{p_i}\\ &= \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^{\nu}}.\end{aligned} \tag{II.86}$$

From (II.84) and (II.86), for all $i \geq i^*$,

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq \max\left\{-\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^{\nu}}\right\}. \tag{II.87}$$

By Proposition II.1(iii), $\|\nabla\psi_{p_i\Omega^*}(x_i)\|$ is bounded because $\{x_i\}_{i=0}^{\infty}$ is bounded. Since $\nu \in (0,1)$, there exists an $i^{**} \in \mathbb{N}_0$, where $i^{**} \geq i^*$, such that

$$-\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i} \geq \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^{\nu}}, \tag{II.88}$$

for all $i \geq i^{**}$. Therefore, from (II.87),

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \tag{II.89}$$

for all $i \geq i^{**}$. Since by Lemma II.19, $\sum_{i=0}^{\infty} 1/p_i = +\infty$, it follows from (II.84) and (II.89) that

$$\psi_{p_i\Omega^*}(x_i) \to -\infty, \text{ as } i \to \infty. \tag{II.90}$$

Let x^* be an accumulation point of $\{x_i\}_{i=0}^{\infty}$. That is, there exists an infinite subset $K \subseteq \mathbb{N}_0$ such that $x_i \to^K x^*$. Based on (II.7), Lemma II.19, and continuity of $\psi_{\Omega^*}(\cdot)$, it follows that $\psi_{p_i\Omega^*}(x_i) \to^K \psi_{\Omega^*}(x^*)$, as $i \to \infty$, which contradicts (II.90). Hence, $\liminf_{i\to\infty}\|\nabla\psi_{p_i\Omega^*}(x_i)\| = 0$. Consequently, there exists an infinite subset $K^* \subseteq \mathbb{N}_0$ and an $x^* \in \mathbb{R}^d$ such that $x_i \to x^*$ and $\theta_{p_i\Omega^*}(x_i) \to^{K^*} 0$, as $i \to \infty$, which implies that $\limsup_{i\to\infty}\theta_{p_i\Omega^*}(x_i) \geq 0$. From Definition I.3, Theorem II.15, and the fact that $\theta_{\Omega^*}(\cdot)$ is a nonpositive function, $\theta_{\Omega^*}(x^*) = 0$. □

Theorem II.22. *Suppose that Assumption II.4(i) holds. (i) If Algorithm II.3 constructs a bounded sequence $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$, then there exists an accumulation point $x^* \in \mathbb{R}^d$ of the sequence $\{x_i\}_{i=0}^{\infty}$ that satisfies $\theta(x^*) = 0$. (ii) If Algorithm II.3 constructs a finite sequence $\{x_i\}_{i=0}^{i^*} \subset \mathbb{R}^d$, then Step 2b constructs an unbounded infinite sequence $\{z_{i^*l'}\}_{l'=l}^{-\infty}$ with*

$$\psi(z_{i^*l'-1}) < \psi(z_{i^*l'}) \tag{II.91}$$

for all $l' \in \{l, l-1, l-2, ...\}$, where l is the tentative Armijo stepsize computed in Step 2a.

Proof. First, we consider (i). Let the set $\Omega^* \subseteq Q$ be as in Lemma II.18, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. Based on Lemma II.21, there exists an accumulation point of the sequence $\{x_i\}_{i=0}^{\infty}$, $x^* \in \mathbb{R}^d$ such that $\theta_{\Omega^*}(x^*) = 0$. The conclusion then follows by similar arguments as in Theorem II.17.

We next consider (ii). Algorithm II.3 constructs a finite sequence only if it jams in Step 2b. Then Substep 1 constructs an infinite sequence $\{z_{i^*l'}\}_{l'=l}^{-\infty}$ satisfying (II.91) for all $l' \in \{l, l-1, l-2, ...\}$. The infinite sequence is unbounded since $h_{p_i\Omega_i}(x_i) \neq 0$ as (II.91) cannot hold otherwise, and $\beta \in (0, 1)$. □

3. Complexity

Next, we consider the complexity in q for a fixed $d \in \mathbb{N}$ of Algorithms II.2 and II.3 to achieve a near-optimal solution of (FMX). Suppose that all functions $f^j(\cdot)$ are active, i.e., $\Omega_i = Q$, near an optimal solution. If $B_{p\Omega}(\cdot)$ is given by (II.47), then the main computational work in each iteration of Algorithms II.2 and II.3 is the calculation of $\nabla\psi_p(\cdot)$, which takes $O(q)$ arithmetic operations under Assumption II.9; see the proof of Theorem II.10. If $B_{p\Omega}(\cdot)$ is given by (II.48), then the main computational work is the calculation of (II.48) and $h_{p\Omega}(x)$. Under Assumption II.9, it takes $O(q)$ arithmetic operations to compute $\mu_p^j(x)$, for all $j \in Q$, $O(q)$ to compute $\nabla f^j(x)$, for all $j \in Q$, $O(q)$ to sum $\sum_{j\in\Omega} \mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T$, $O(q)$ to sum $\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)$, and the other operations take $O(1)$. In all, the number of arithmetic operations to obtain $B_{p\Omega}(x)$ is $O(q)$. A direct method for solving a linear system of equations to compute $h_{p\Omega}(x)$ depends on d, but is constant in q. Hence, if $B_{p\Omega}(\cdot)$ is given by (II.48), the computational work in each iteration of Algorithms II.2 and II.3 is $O(q)$. It is unclear how many iterations Algorithms II.2 and II.3 would need to achieve a near-optimal solution as a function of q. However, since they may utilize Quasi-Newton search directions and adaptive precision adjustment, there is reason to believe that the number of iterations will be no larger than that of Algorithm II.1, which uses the steepest descent direction and a fixed precision parameter. Thus, suppose that for some tolerance $t > 0$, the number of iterations of Algorithms II.2 and II.3 to generate

$\{x_i\}_{i=0}^n$, with the last iterate satisfying $\psi(x_n) - \psi^* \leq t$, is no larger than $O(\log q)$, as is the case for Algorithm II.1. Then the complexity of Algorithms II.2 and II.3 to generate x_n is no larger than $O(q \log q)$, which is the same as for Algorithm II.1.

E. NUMERICAL RESULTS

We present an empirical comparison of Algorithms II.2 and II.3 with algorithms from the literature over a set of problem instances from Polak et al. (2003); Zhou and Tits (1996) as well as randomly generated instances; see Appendix A. This study appears to be the first systematic comparison of smoothing and SQP algorithms for large-scale problems, with number of functions q up to two orders of magnitude larger than previously reported. Specifically, we examine:

(i) PPP. Pshenichnyi-Pironneau-Polak min-max algorithm (Algorithm 2.4.1 in Polak 1997).

(ii) ϵ-PPP. An active-set version of PPP as stated in Algorithm 2.4.34 in Polak (1997); see also Polak (2008).

(iii) SQP-2QP. Algorithm 2.1 of Zhou and Tits (1996), an SQP algorithm with two QPs.

(iv) SQP-1QP. Algorithm A in Zhu et al. (2009), a one-QP SQP algorithm.

(v) SMQN. Algorithm 3.2 in Polak et al. (2008), a smoothing Quasi-Newton algorithm.

(vi) Algorithms II.2 and II.3 of the present chapter.

We refer to Appendix B for details about algorithm parameters. With the exception of PPP and SQP-1QP, the above algorithms incorporate active-set strategies and, hence, appear especially promising for solving problem instances with large q. We implement and run all algorithms in MATLAB version 7.7.0 (R2008b) (see Mathworks 2009) on a 3.73 GHz PC using Windows XP SP3, with 3 GB of RAM. All QPs are solved using TOMLAB CPLEX version 7.0 (R7.0.0) (see Tomlab 2009) with the Primal Simplex option, which preliminary studies indicate result in the smallest QP

run time. We also examined the LSSOL QP solver (see Gill, Hammarling, Murray, Saunders, & Wright, 1986), but its run times appear inferior to that of CPLEX for large-scale QPs arising in the present context.

Algorithm 2.1 of Zhou and Tits (1996) is implemented in the solver CFSQP (Lawrence, Zhou, & Tits, 1997) and we have verified that our MATLAB implementation of that algorithm produces comparable results in terms of number of iterations and run time as CFSQP. We do not directly compare with CFSQP as we find it more valuable to compare different algorithms using the same implementation environment (MATLAB) and the same QP solver (CPLEX).

For Algorithm II.3, unless otherwise stated, we use the Quasi-Newton direction with $B_{p\Omega}(x)$ as defined in (II.48), because preliminary test runs show that generally, the alternate steepest descent direction with $B_{p\Omega}(x)$ as defined in (II.47) produces longer run times. We examine all problem instances from Polak et al. (2003); Zhou and Tits (1996) except two that cannot be easily extended to large q. As the problem instances with many variables in Polak et al. (2003); Zhou and Tits (1996) do not allow us to adjust the number of functions, we create two additional sets of problem instances; see Appendix A for details. We report run times to achieve a solution x that satisfies

$$\psi(x) - \psi^{\text{target}} \leq t, \tag{II.92}$$

where ψ^{target} is a target value (see Table 17 of Appendix A) equal to the optimal value (if known) or a slightly adjusted value from the optimal values reported in Polak et al. (2003); Zhou and Tits (1996) for smaller q. We use $t = 10^{-5}$. Although this termination criteria is not possible for real-world problems, we find that it is the most useful criterion in this study.

Before we can compare the run times of the various algorithms, we need to conduct sensitivity analysis to determine a robust setting (one that produces the fastest run times for majority of the problem instances) for the parameter ϵ (see (II.46)) to use for the active-set strategies.

1. Selection of a Robust ϵ for Active-Set Algorithms

Of the algorithms compared, ϵ-PPP, SQP-2QP, SMQN, Algorithms II.2 and II.3 implement some form of active-set strategies. The performance of these active-set algorithms depend on the parameter ϵ, which defines an ϵ-active set at each iteration. However, as ϵ is not used exactly the same way in the different algorithms, we do not expect the robust ϵ setting to be the similar for the different algorithms.

For the sensitivity analysis, we use the same set of problem instances ProbC, ProbG, and ProbL (see Appendix A) for all active-set algorithms. The three problem instances have different problem dimensionality d, which we hope contribute a robust setting for ϵ. We include the non-convex ProbG (see Table 17 of Appendix A) to ensure that the chosen ϵ is robust for both convex and non-convex problems.

The number of objective functions for each test problem, q is set as high as possible (in powers of 10), without encountering memory problems for any of the algorithms. For each problem instance and active-set algorithm, we determine the run times with $\epsilon = 1000, 100, ..., 10^{-20}$. We present a representative sample of the run times, leaving out (i) those run times that do not change much when we decrease ϵ by a factor of 10, and (ii) those run times that are significantly longer than the fastest run time.

a. Selection of a Robust ϵ for ϵ-PPP

Table 1 indicates that the performance of the algorithm ϵ-PPP is sensitive to ϵ, and there is no single value of ϵ that is consistently better for the three problem instances considered. The word "local" indicates that the algorithm converges to a locally optimal solution for the non-convex ProbG. The run times with $\epsilon = 10^{-2}$ to $\epsilon = 10^{-4}$ seem to be consistently better than other settings, and we will use $\epsilon = 10^{-3}$ for the algorithm comparison study.

b. Selection of a Robust ϵ for SQP-2QP

Table 2 indicates that the performance of the algorithm SQP-2QP is relatively insensitive to different ϵ values. We use $\epsilon = 1$ for the algorithm comparison,

	ProbC	ProbG	ProbL
d	2	4	4×10^2
q	10^5	10^5	10^2
$\epsilon = 1000$	869.7	local	315.1
$\epsilon = 100$	540.5	local	243.3
$\epsilon = 10$	350.9	local	190.9
$\epsilon = 1$	71.7	local	140.8
$\epsilon = 10^{-1}$	35.5	local	101.0
$\epsilon = 10^{-2}$	5.1	local	79.2
$\epsilon = 10^{-3}$	5.0	local	79.7
$\epsilon = 10^{-4}$	3.1	local	104.9
$\epsilon = 10^{-5}$	3.1	local	197.1
$\epsilon = 10^{-10}$	31.5	local	4246
$\epsilon = 10^{-15}$	> 7200	local	> 7200
$\epsilon = 10^{-20}$	> 7200	local	> 7200

Table 1. Run times based on ϵ for ϵ-PPP. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems.

	ProbC	ProbG	ProbL
d	2	4	4×10^2
q	10^5	10^5	10^2
$\epsilon = 1000$	1.7	2.7	21.5
$\epsilon = 100$	0.85	2.4	21.4
$\epsilon = 10$	0.74	2.5	21.4
$\epsilon = 1$	0.67	2.4	15.1
$\epsilon = 10^{-1}$	0.71	2.5	15.0
$\epsilon = 10^{-5}$	0.76	3.2	14.3
$\epsilon = 10^{-10}$	0.72	3.2	14.2
$\epsilon = 10^{-15}$	0.76	3.2	14.4
$\epsilon = 10^{-20}$	0.68	3.1	14.3

Table 2. Run times based on ϵ for SQP-2QP.

as it provides consistently fast run times as seen in Table 2, and it is also the proposed value in Zhou and Tits (1996).

c. Selection of a Robust ϵ for SMQN and Algorithm II.2

Algorithm II.2 is very similar to SMQN, the only difference being the schemes for precision-parameter adjustment. Due to their similarity, we conduct the sensitivity analysis with only SMQN, but apply the resulting ϵ to both algorithms for the algorithm comparison.

	ProbC	ProbG	ProbL
d	2	4	4×10^2
q	10^5	10^5	10^2
$\epsilon = 1000$	152.6	105.2	584.8
$\epsilon = 100$	152.5	105.5	571.3
$\epsilon = 10$	153.0	103.6	845.3
$\epsilon = 1$	140.0	116.5	547.8
$\epsilon = 10^{-1}$	112.0	108.2	153.2
$\epsilon = 10^{-5}$	83.9	216.3	113.9
$\epsilon = 10^{-10}$	11.8	31.2	113.9
$\epsilon = 10^{-15}$	12.2	29.8	114.1
$\epsilon = 10^{-20}$	12.6	25.3	114.0

Table 3. Run times based on ϵ for SMQN and Algorithm II.2.

Table 3 provides a clear indication that a small ϵ provides the fastest run times for SMQN consistently. There is no recommended setting for the parameter ϵ in Polak et al. (2008). We select $\epsilon = 10^{-20}$ for SMQN and Algorithm II.2 for the algorithm comparison.

d. Selection of a Robust ϵ for Algorithm II.3

Table 4 indicates that the performance of Algorithm II.3 is sensitive to the value of ϵ and there is not a single ϵ value that is optimal for the three problem instances selected. Similar to SMQN and Algorithm II.2, we use $\epsilon = 10^{-20}$ for the algorithm comparison.

	ProbC	ProbG	ProbL
d	2	4	4×10^2
q	10^5	10^5	10^2
$\epsilon = 1000$	5.4	local	0.34
$\epsilon = 100$	5.4	local	0.35
$\epsilon = 10$	3.7	local	0.34
$\epsilon = 1$	4.3	local	0.77
$\epsilon = 10^{-1}$	3.0	local	3.4
$\epsilon = 10^{-5}$	3.5	557.4	4.3
$\epsilon = 10^{-10}$	0.96	27.6	4.2
$\epsilon = 10^{-15}$	1.2	22.3	4.1
$\epsilon = 10^{-20}$	1.3	20.1	4.6

Table 4. Run times based on ϵ for Algorithm II.3. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems.

In view of the above sensitivity analyses, we use the following values of ϵ to compare the various algorithms in the next section, $\epsilon = 10^{-3}$ for ϵ-PPP, $\epsilon = 1$ for SQP-2QP, and $\epsilon = 10^{-20}$ for SMQN, Algorithms II.2 and II.3.

2. Comparison

In this subsection, we compare the algorithms over a set of problem instances from Polak et al. (2003); Zhou and Tits (1996) as well as randomly generated instances; see Appendix A.

a. Minimizing the Maximum of up to 100,000 Functions

Table 5 summarizes the run times (in seconds) of the various algorithms, with Columns 2 and 3 giving the number of variables d and functions q, respectively. Run times in boldface indicate that the particular algorithm has the shortest run time for the specific problem instance. The numerical results in Table 5 indicate that in most problem instances, the run times are shortest for SQP-2QP or Algorithm II.3. Table 5 indicates that SQP-2QP is significantly more efficient than SQP-1QP for problem instances ProbA-ProbG. This is due to the efficiency of the active-set strategy

in SQP-2QP, which is absent in SQP-1QP. However, for ProbJ-ProbM, SQP-1QP is comparable to SQP-2QP. This is because at the optimal solution of ProbJ-ProbM, all the functions are active. This causes the active-set strategy in SQP-2QP to lose its effectiveness as the optimal solution is approached.

Table 5 indicates also that Algorithm II.2 is more efficient than SMQN for most problem instances. As the only difference between the two algorithms lies in their precision-parameter adjustment scheme, this highlights the sensitivity in the performance of smoothing algorithms to the control of their precision parameters. Table 5 also shows that Algorithm II.3 is more efficient than Algorithm II.2 and SMQN for most problem instances.

Table 5 indicates that SQP-2QP is generally more efficient than Algorithm II.3 for problem instances with small dimensionality, $d \leq 4$ (specifically ProbA-ProbG), and vice versa. This is consistent with the common observation that SQP-type algorithms may be inefficient for problems with many variables; see for example Zhou and Tits (1996).

Table 5 shows that some algorithms return locally optimal solutions for some problem instances (labeled "local" in Table 5). In view of these results, there is an indication that smoothing algorithms (SMQN, Algorithms II.2 and II.3) tend to find global minima more frequently than PPP and SQP algorithms.

Instance	d	q	PPP	ϵ-PPP ($\epsilon = 10^{-3}$)	SQP-2QP ($\epsilon = 1$)	SQP-1QP	SMQN ($\epsilon = 10^{-20}$)	Algo II.2 ($\epsilon = 10^{-20}$)	Algo II.3 ($\epsilon = 10^{-20}$)
ProbA	1	100,000	17.3	2.5	0.45	13.7	0.64	0.41	**0.31**
ProbB	1	100,000	2.5	0.69	**0.06**	131.1	0.31	0.70	0.45
ProbC	2	100,000	15.3	5.0	**0.67**	11.9	12.6	1.9	1.3
ProbD	2	100,000	5.0	7.4	**0.21**	9.9	7.2	1.7	1.5
ProbE	3	100,000	19.5	14.5	**0.59**	18.3	8.1	2.2	2.0
ProbF	3	100,000	28.7	18.8	2.3	24.4	18.2	2.9	**2.1**
ProbG	4	100,000	local	local	**2.4**	79.1	25.3	28.7	20.1
ProbH	4	100,000	211.8	968.4	local	128.9	36.2	31.5	**23.5**
ProbI	6	100,000	37.7	local	local	**31.7**	425.2	512.9	local
ProbJ	1,000	1,000	*	*	1458	1161	**	2212	**465.6**
ProbK	2,000	1,000	*	*	**	8404	**	10620	**2265**
ProbL	400	100	**3.6**	79.7	15.1	14.5	112.0	17.5	4.6
ProbM	100	4,950	7.0	160.3	2.7	3.6	4.5	3.6	**2.3**

Table 5. Run times (in seconds) for various algorithms. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (II.92) after six hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 10^{-3}$ at termination.

b. Minimizing the Maximum of up to 1,000,000 Functions

Table 6 presents similar results as in Table 5, but for larger q. We do not present results for PPP and SQP-1QP as the required QPs exceed the memory limit. The comprehensive sensitivity studies for ϵ show significant improvement for Algorithm II.3 for ProbJ-ProbM if a large ϵ is used. Hence, we include the results for Algorithm II.3 with $\epsilon = 1000$ in Table 6. This ϵ-value means that there is effectively no active-set strategy. Sensitivity tests conducted for the other algorithms with a larger ϵ show no improvement in their run times.

The observations from Table 6 are similar to those for Table 5. Table 6 indicates that Algorithm II.3 with $\epsilon = 1000$ is efficient for ProbJ-ProbM, which has large d and a significant number of functions active at the optimal solution. For completeness, the run times for Algorithm II.3 with $\epsilon = 1000$ for ProbJ-ProbM in Table 5 are 2.8, 14.3, 0.36 and 3.0 seconds respectively, while the run times for the other problem instances are longer than Algorithm II.3 with $\epsilon = 10^{-20}$.

The results in Tables 5 and 6 indicate that among the algorithms considered, SQP-2QP and Algorithm II.3 are the most efficient algorithms for minimax problems with a large number of functions. The run times for ProbJ-ProbM indicate that SQP-2QP is less efficient for problem instances with a significant number of the functions that is nearly active at the solution, as the active-set strategy loses its effectiveness.

Instance	d	q	ϵ-PPP ($\epsilon = 10^{-3}$)	SQP-2QP ($\epsilon = 1$)	SMQN ($\epsilon = 10^{-20}$)	Algo II.2 ($\epsilon = 10^{-20}$)	Algo II.3 ($\epsilon = 1000$)	Algo II.3 ($\epsilon = 10^{-20}$)
ProbA	1	1,000,000	22.5	4.6	4.4	**2.7**	8.6	3.1
ProbB	1	1,000,000	6.1	**0.61**	2.7	4.8	2.5	3.0
ProbC	2	1,000,000	59.4	**7.2**	131.0	15.0	61.9	13.1
ProbD	2	1,000,000	79.3	**2.2**	75.3	12.3	47.5	13.4
ProbE	3	1,000,000	245.0	**5.5**	93.0	12.1	74.5	17.5
ProbF	3	1,000,000	332.9	22.9	185.9	21.7	74.1	**18.1**
ProbG	4	1,000,000	local	**27.2**	257.8	220.1	12227	169.5
ProbH	4	1,000,000	12322	local	362.8	240.4	4157	**238.4**
ProbI	6	1,000,000	local	local	4262	4016	**3717**	local
ProbJ	4,000	4,000	**	**	**	**	**92.8**	**
ProbK	4,000	2,000	**	**	**	**	**91.8**	**
ProbL	4,000	1,000	*	**	**	**	**106.6**	13273
ProbM	200	19,900	*	24.7	66.1	917.3	8.6	**2.5**

Table 6. Similar results as in Table 5, but with larger q. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (II.92), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (II.92) after six hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 0.01$ at termination.

c. Randomly Generated Problem Instances

The problem instances from the literature examined in Tables 5 and 6 include either cases with few functions ϵ-active at an optimal solution (ProbA-ProbI) or cases with all functions ϵ-active (ProbJ-ProbM). We also examine randomly generated problem instances with an intermediate number of functions ϵ-active at the optimal solution; see ProbN in Table 17 of Appendix A. The optimal values are unknown in this case but the target values given in Table 17 of Appendix A appear to be close to the global minima.

d	q	SQP-2QP ($\epsilon = 1$)	Algo II.3 SD ($\epsilon = 1000$)	Algo II.3 QN ($\epsilon = 1000$)
10	10,000	**0.42**	0.64	0.62
100	10,000	0.82	**0.48**	0.54
1,000	10,000	124.9	**0.38**	4.8
10	100,000	4.1	**3.8**	4.2
100	100,000	11.5	**3.8**	4.1
1,000	100,000	mem	**4.3**	9.7
1,000	1,000,000	mem	**37.2**	42.5
1,000	10,000,000	mem	**421.8**	492.5
10,000	100,000	mem	**6.3**	mem

Table 7. Run times (in seconds) of algorithms on problem instance ProbN. "SD" and "QN" indicate that Algorithm II.3 uses $B_{p\Omega}(\cdot)$ given by (II.47) and (II.48), respectively. The word "mem" indicates that the algorithm terminates due to insufficient memory.

Table 7 presents the run times for Algorithm II.3 and SQP-2QP on ProbN. As the problem instances are relatively well-conditioned, Algorithm II.3 with $B_{p\Omega}(\cdot)$ given by (II.47), i.e., a steepest descent (SD) direction, may perform well and is included in the table. The parameter ϵ for Algorithm II.3 is set to 1000 for this set of problem instances, as preliminary test runs show that it is consistently better than other choices. Table 7 indicates that SQP-2QP is less efficient than Algorithm II.3 for problem instances with large d, and where there is a significant number of functions ϵ-active at the optimal solution. The last row in Table 7 shows that for

problem instances with $d \geq 10,000$, the storage of the $d \times d$ $H_{p\Omega}(\cdot)$ matrix for both SQP-2QP and Algorithm II.3, with $B_{p\Omega}(\cdot)$ given by (II.48), causes both algorithms to terminate due to memory limitations. Thus, Algorithm II.3, with $B_{p\Omega}(\cdot)$ given by (II.47), which does not have any matrix to store, may be a reasonable alternative when d is large.

F. CONCLUSIONS FOR FINITE MINIMAX

This chapter focuses on minimizing the maximum of many functions and presents complexity and rate-of-convergence analysis of smoothing algorithms for such problems. We find that smoothing algorithms might only have sublinear rates of convergence, but their complexity in the number of functions is competitive with other algorithms due to small computational work per iteration. We present two smoothing algorithms with novel precision-adjustment schemes and carry out a comprehensive numerical comparison with other algorithms from the literature. We find that the proposed algorithms are more efficient than a recent smoothing algorithm from the literature, due to the more efficient precision-adjustment schemes implemented. The proposed algorithms are competitive with SQP algorithms, and especially efficient for problem instances with many variables, or where a significant number of functions are nearly active at stationary points. The numerical results indicate that smoothing with first-order gradient methods is likely the only viable approach to solve finite minimax problems with many functions and variables due to memory issues.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SEMI-INFINITE MINIMAX PROBLEM

A. INTRODUCTION

In this chapter, we consider semi-infinite minimax problems of the form

$$\text{(SMX)} \quad \min_{x \in \mathbb{R}^d} \psi(x), \tag{III.1}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\psi(x) \triangleq \max_{y \in Y} \phi(x, y), \tag{III.2}$$

Y is a compact infinite subset of $\mathbb{R}^m$, $\phi : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$, $d, m \in \mathbb{N}$, and $\phi(\cdot, \cdot)$ is continuous and sufficiently smooth on $\mathbb{R}^d \times Y$ as specified below. Note that if Y is a finite set instead, we have the finite minimax problem (FMX) from Chapter II. The notation in each chapter is self-contained. Hence, we here as well as below reuse some symbols from Chapter II in definitions of new quantities. The data m, i.e., the dimension of y, is as we see below a key quantity and we refer to as the *uncertainty dimension*.

In general, (SMX) is used by decision makers to determine the optimal response to the worst-case scenario. (SMX) arises in applications such as finance (Rustem & Howe, 2002), electrical circuit theory (Demyanov & Malozemov, 1974), and policy optimization (Becker, Dwolatzky, Karakitsos, & Rustem, 1986). Solving (SMX) is difficult for two reasons: (i) for any $x \in \mathbb{R}^d, \psi(x)$ may not be computable in finite time because of the global maximization involved, and (ii) $\psi(\cdot)$ may not be differentiable everywhere.

Several methods have been proposed to solve (SMX); see Rustem and Howe (2002, Chapter 2) for a survey of semi-infinite minimax algorithms. A key method for solving (SMX) is the use of semi-infinite programming (SIP) methods. (SMX) can be reformulated into the SIP

$$\text{(SMX}') \quad \min_{(x,z) \in \mathbb{R}^{d+1}} \{z \mid \phi(x, y) - z \leq 0 \ \ \forall y \in Y\}, \tag{III.3}$$

involving an infinite number of constraints, which can then be solved by any SIP algorithm. SIPs are usually solved by solving a sequence of finite problems, i.e., problems with a finite number of constraints. Depending on how the finite problems are created, we can generally group SIP algorithms into three classes: exchange algorithms, local reduction algorithms, and discretization algorithms; see Lopez and Still (2007); Hettich and Kortanek (1993); Reemtsen and Gorner (1998) for surveys on the theory, applications and algorithms of SIP.

In exchange algorithms (Kortanek & No, 1993), at each iterate $(x_i, z_i) \in \mathbb{R}^{d+1}$, $i \in \mathbb{N}$, new constraints $\phi(x, \hat{y}_i) - z \leq 0$ corresponding to a maximizer $\hat{y}_i \in \arg\max_{y \in Y} \phi(x_i, y)$ are added to the finite problem, and existing constraints removed, i.e., an exchange of constraints occurs. In local reduction algorithms (Price & Coope, 1990), under certain regularity assumptions, the SIP can be converted locally into a finite problem.

Discretization algorithms are one of the more popular classes of algorithms for solving SIPs due to their simplicity. They create finite problems by considering a finite discretized subset of Y. To achieve the required solution tolerance, most discretization algorithms implement some kind of adaptive discretization refinement rule to gradually increase the level of discretization, rather than fix the discretization at a high level right from the start. In this chapter, we refer to those algorithms that are applied to solve the individual discretized problems as *algorithm maps*, to differentiate them from the overall discretization algorithm that usually includes some adaptive discretization refinement rule. At each stage of the algorithm, the level of discretization is fixed and an algorithm map is used to solve the finite problem approximately. The approximate solution is then usually used to warm-start the next stage. We refer to Hettich (1986); Reemtsen (1991); Polak and He (1992); Polak (1997) for examples of discretization algorithms.

There are also algorithms that directly address (SMX) without the reformulation to SIP. The algorithms in Chaney (1982); Klessig and Polak (1973) assume

that the maximum in (III.2) occurs at a unique point $\hat{y}(x)$, for all $x \in \mathbb{R}^d$, which ensures that $\psi(\cdot)$ is differentiable. Smooth optimization methods, such as the method of centers algorithm in Chaney (1982) and a first-order, feasible directions method in Klessig and Polak (1973), are then used to minimize the smooth $\phi(\cdot, \hat{y}(\cdot))$. The conceptual algorithm in Panin (1981) and an implementable version in Kiwiel (1987) use a convex piecewise linear approximation of $\psi(\cdot)$ to solve (SMX). As (SMX) belongs to the general class of nonsmooth problems, nonsmooth optimization algorithms such as subgradient and bundle algorithms (Rustem & Howe, 2002) can be used as well. Subgradient algorithms determine the descent direction by computing at least one subgradient at each iterate, while bundle algorithms use subgradient information over several successive iterates to determine the descent direction. A discretization algorithm that does not involve the reformulation into a SIP is proposed in Demyanov and Malozemov (1971). The algorithm solves an infinite sequence of finite minimax problems of the form

$$\min_{x \in \mathbb{R}^d} \max_{y \in Y_N} \phi(x, y), \tag{III.4}$$

where Y_N, $N \in \mathbb{N}$, are finite discretized subsets of Y. This approach is fundamentally the same as converting (SMX) into a SIP and then applying discretization methods.

In this chapter, we propose a novel way of expressing rate of convergence, in terms of computational work instead of the typical number of iterations. We first discuss the inadequacy of the typical rate of convergence. We consider two adaptive discretization algorithms (Polak & He, 1992; Polak, Mayne, & Higgins, 1992) to solve (SMX). Polak and He (1992) propose a set of discretization refinement rules, which ensures that their adaptive discretization algorithm generates sequences that converge to a solution of the original SIP problem at the same linear rate with the same estimated rate constant as that of the linearly convergent algorithm map used in the discretization algorithm. Another similar study that investigates this rate-preserving idea is found in Polak et al. (1992) for a semi-infinite minimax algorithm, which uses an extension to Newton's method as the algorithm map. Polak et al. (1992) state

that the rate of convergence for their adaptive discretization algorithm is superlinear. Without further information, a user probably will select the superlinearly convergent algorithm to solve (SMX). However, for the superlinearly convergent algorithm, because the level of discretization needs to increase rapidly to achieve the superlinear rate, the computational work between iterates increases rapidly. Thus, the computational time may not be well-correlated to the superlinear rate of convergence since the typical rate of convergence does not consider computational work.

To our knowledge, there has been no rate-of-convergence result that considers computational work for discretization algorithms for SIP and (SMX). That said, not all rate-of-convergence results are in terms of the number of iterations. Still (2001) studies how the rate of convergence for SIP discretization algorithms depends on the level of discretization and whether the discretization includes boundary points of Y in a specific way. Shapiro (2009) determines the rate of convergence of an ϵ-optimal solution of the discretized problem to the set of optimal solutions of the SIP problem, as a function of the level of discretization.

In our proposed way of expressing rate of convergence, we relate computational work to the number of iterations as well as to the level of discretization by making some computational work assumptions. This relation allows us to determine the rate of decay of a bound on the error between the iterates generated from the discretized problems and the optimal solution of (SMX) as a function of computational work, which we refer to as *rate of decay of error bound* in the rest of the chapter. We use this new way to develop rate-of-convergence results for various fixed and adaptive discretization algorithms for (SMX) and compare them against the rate of convergence of an ϵ-subgradient algorithm. We show that the new way allows a fairer comparison of the various algorithms than the typical rate of convergence. We also conduct numerical studies to validate the theoretical results we obtain.

The next section describes the discretization approach and determines the rate of decay of error bound for discretization algorithms using algorithm maps with

varying rate of convergence. Section C determines the rate of decay of error bound for an ϵ-subgradient algorithm and compares it against the discretization algorithms. Section D contains numerical results.

B. EFFICIENCY OF DISCRETIZATION ALGORITHM

We start this section by describing the discretization approach for (SMX) and include for completeness some known results that we use in later subsections.

1. Discretization

The discretization approach involves approximating Y by a finite subset $Y_N \subset Y$, where $|Y_N| = N$ ($|\cdot|$ denotes the cardinality operator), and approximately solving the resulting finite minimax problem

$$(\text{SMX}_N) \quad \min_{x \in \mathbb{R}^d} \psi_N(x), \tag{III.5}$$

where $\psi_N : \mathbb{R}^d \to \mathbb{R}$, $N \in \mathbb{N}$, is defined by

$$\psi_N(x) \stackrel{\triangle}{=} \max_{y \in Y_N} \phi(x, y). \tag{III.6}$$

(SMX_N) can be solved using any finite minimax algorithms, such as those in Chapter II. In the remainder of this chapter, we refer to elements of Y_N as *grid points*. When they exist, we denote the optimal solutions of (SMX) and (SMX_N) by x^* and x_N^*, respectively, and the corresponding optimal values by ψ^* and ψ_N^*. We next state some properties of $\psi(\cdot)$ and $\psi_N(\cdot)$.

Proposition III.1. *The following facts hold:*

(i) For all $x \in \mathbb{R}^d$ and $N \in \mathbb{N}$, $\psi_N(x) \leq \psi(x)$.

(ii) Suppose that $\phi(\cdot, \cdot)$ is continuous on $\mathbb{R}^d \times Y$. Then $\psi(\cdot)$ and $\psi_N(\cdot)$ are continuous for any $N \in \mathbb{N}$ on $\mathbb{R}^d$.

(iii) For all $N \in \mathbb{N}$,

$$\psi_N^* \leq \psi^*. \tag{III.7}$$

Proof. The conclusion (i) follows directly from the definitions of $\psi(\cdot)$ and $\psi_N(\cdot)$, and the fact that $Y_N \subset Y$. Part (ii) follows, for example, from pp. 51 and 187 of Demyanov and Malozemov (1974). For part (iii), by definition of x_N^*, $\psi_N(x_N^*) \leq \psi_N(x)$ for all $x \in \mathbb{R}^d$ (which includes x^*), thus, based on part (i),

$$\psi_N^* \stackrel{\triangle}{=} \psi_N(x_N^*) \leq \psi_N(x^*) \leq \psi(x^*) \stackrel{\triangle}{=} \psi^*. \tag{III.8}$$

□

In this section, we focus on the following basic fixed discretization algorithm, for which we develop a series of rate of decay of error bound results.

Algorithm III.1. Fixed Discretization Algorithm
Data: $x_0 \in \mathbb{R}^d$.

Parameters: Discretization parameter $N \in \mathbb{N}$ and parameters required for the algorithm map.

Step 1. Generate a sequence $\{x_i\}_{i=0}^{\infty}$ by applying an algorithm map to (SMX_N). □

We need the following assumptions for the rate of decay of error bound analysis. The operator $\|\cdot\|$ denotes the Euclidean norm.

Assumption III.2. *The functions $\phi(x,\cdot), x \in \mathbb{R}^d$, are uniformly Lipschitz continuous in y, i.e., there exists a constant $L < \infty$ such that*

$$|\phi(x,y) - \phi(x,y')| \leq L\|y - y'\| \tag{III.9}$$

for all $x \in \mathbb{R}^d$ and $y, y' \in \mathbb{R}^m$. □

We require an assumption on the discretization scheme, which dictates how Y_N is generated from Y given a $N \in \mathbb{N}$. We assume that the same discretization scheme is used throughout this chapter for the various algorithms.

Assumption III.3. *There exists a $N_1 \in \mathbb{N}$, a discretization scheme defined for all $N \in \mathbb{N}, N \geq N_1$, and a monotonically decreasing function $\Delta_m : \mathbb{N} \to \mathbb{R}$, where m is the dimensionality of Y and $\Delta_m(N) \to 0$ as $N \to \infty$, such that*

$$0 \leq \psi(x) - \psi_N(x) \leq \Delta_m(N) \tag{III.10}$$

for all $x \in \mathbb{R}^d$ *and* $N \in \mathbb{N}, N \geq N_1$. *In addition, there exists a* $L' < \infty$ *such that the discretization error* $\Delta_m(N)$ *can be expressed as*

$$\Delta_m(N) \triangleq L' \frac{\sqrt{m}}{N^{1/m}} \tag{III.11}$$

for all $N \in \mathbb{N}, N \geq N_1, m \in \mathbb{N}$. □

Under Assumption III.2, Assumption III.3 holds for example when Y is the unit cube $[0,1]^m$, and $Y_N, N > 2^m$, is the uniform grid I_N^m defined in each of m dimensions by

$$I_N \triangleq \left\{0, \frac{1}{\lfloor N^{1/m} \rfloor - 1}, \frac{2}{\lfloor N^{1/m} \rfloor - 1}, ..., 1\right\} \tag{III.12}$$

and $\lfloor \cdot \rfloor$ denotes the floor function.

There are $\lfloor N^{1/m} \rfloor$ grid points in each of the m dimensions of Y, for a total of $\lfloor N^{1/m} \rfloor^m$ grid points. Thus, each grid element is a cube with length $\frac{1}{\lfloor N^{1/m} \rfloor - 1}$ for each edge of the grid element.

To continue the discussion, we need a way to quantify the "distance" between two sets. We use Hausdorff distance for this purpose. The Hausdorff distance between Y and Y_N is defined as

$$\text{dist}(Y, Y_N) \triangleq \max_{y \in Y} \min_{y' \in Y_N} \|y' - y\|. \tag{III.13}$$

The Hausdorff distance between Y and Y_N is the maximum distance between any point $y \in Y$ and its nearest grid point in Y_N.

For the unit cube example, the Hausdorff distance between Y and Y_N is then the distance from the center to a corner of the grid element, which, based on the Euclidean distance of two points in m-dimensional space, is

$$\text{dist}(Y, Y_N) = \frac{\sqrt{m}}{2(\lfloor N^{1/m} \rfloor - 1)}. \tag{III.14}$$

Let $\hat{y} \in \hat{Y}(x) \triangleq \arg\max_{y \in Y} \phi(x, y)$, and $y_1 \in Y_N$ be the nearest grid point to $\hat{y}$. Based on the definition of the Hausdorff distance,

$$\|y_1 - \hat{y}\| \leq \frac{\sqrt{m}}{2(\lfloor N^{1/m} \rfloor - 1)}. \tag{III.15}$$

Under Assumption III.2, there exists a constant $L < \infty$ such that

$$\phi(x, \hat{y}) - \phi(x, y_1) \leq L\|y_1 - \hat{y}\|. \tag{III.16}$$

Let $\hat{Y}_N(x) \stackrel{\triangle}{=} \arg\max_{y \in Y_N} \phi(x, y)$ and $y_2 \in \hat{Y}_N(x)$. Thus, $\phi(x, y_2) \geq \phi(x, y_1)$ and

$$\phi(x, \hat{y}) - \phi(x, y_2) \leq \phi(x, \hat{y}) - \phi(x, y_1). \tag{III.17}$$

Since $\psi(x) = \phi(x, \hat{y})$ and $\psi_N(x) = \phi(x, y_2)$ by definition, for $N > 2^m$,

$$0 \leq \psi(x) - \psi_N(x) \leq L\frac{\sqrt{m}}{2(\lfloor N^{1/m} \rfloor - 1)} \leq L\frac{\sqrt{m}}{2(N^{1/m} - 2)}. \tag{III.18}$$

There exists a $N_1 \in \mathbb{N}$ such that

$$L\frac{\sqrt{m}}{2(N^{1/m} - 2)} \leq L\frac{\sqrt{m}}{N^{1/m}} \tag{III.19}$$

for all $N \geq N_1$. This completes the verification that Assumption III.3 holds for the unit cube with a uniform grid.

We need the following strong convexity assumption, which is standard for rate-of-convergence analysis; see for example Polak et al. (1992).

Assumption III.4. *The function $\phi(\cdot, y)$, for all $y \in Y$, is twice continuously differentiable, and there exists an $a \in (0, \infty)$ such that*

$$a\|z\|^2 \leq \langle z, \nabla^2_{xx}\phi(x, y)z\rangle, \tag{III.20}$$

for all $x, z \in \mathbb{R}^d$, and $y \in Y$. □

In the following subsections, we derive the rate of decay of error bounds of fixed (Algorithm III.1) and adaptive (Algorithm III.2) discretization algorithms to solve (SMX) in terms of computational work. Hence, we need to define precisely what we mean by an error bound for the various algorithms. For fixed discretization algorithms (Algorithm III.1), we denote the n^{th} iterate of a fixed discretization algorithm based on discretization parameter N by x_n^N. Suppose that a computational budget $b \in (0, \infty)$

is allocated to solve (SMX), and the computational work required to run $n_b \in \mathbb{N}$ iterations of a fixed discretization algorithm on (SMX_{N_b}), $N_b \in \mathbb{N}$, is no larger than b. We refer to the quantity $\psi\left(x_{n_b}^{N_b}\right) - \psi^*$ as the *error*. An upper bound on this quantity is referred to as an *error bound*. Obviously, there are many possible error bounds. We will define several specific error bounds for analysis.

For the rate of decay of error bound analyses in this section, we consider a fixed discretization algorithm, Algorithm III.1, with an *ideal* algorithm map that solves (SMX_{N_b}), $N_b \in \mathbb{N}$, exactly in one iteration; we consider an adaptive discretization algorithm, Algorithm III.2, and we consider Algorithm III.1 with algorithm maps with quadratic, linear, sublinear rate of convergence, as well as a specific case of a smoothing algorithm.

We need the following assumption on computational work and budget for the rate analysis.

Assumption III.5. *There exist $\sigma \in (0, \infty)$ and $\nu \in [1, \infty)$ such that the computational work required in each iteration of the algorithm map in solving* (SMX_N) *is no larger than σN^ν for all $N \in \mathbb{N}$.* □

The preceding assumption holds with $\nu = 1$ for the two smoothing algorithms proposed in Chapter II, and holds with $\nu = 3$ for the SQP and PPP algorithms discussed in Chapter II. Suppose that the assumption holds for the algorithm map under consideration and a computational budget of $b \in \mathbb{N}$ is allocated to Algorithm III.1, to run n_b iterations of the algorithm map on (SMX_{N_b}). Then N_b and n_b must be picked such that

$$\sigma N_b^\nu n_b \leq b. \tag{III.21}$$

In the upcoming analyses, we see that the error bounds for the various algorithms often have two components. The first component is the error of not achieving the optimal solution of the discretized (SMX_{N_b}), which decreases monotonically as the number of iterations n_b increase. The second component of the error bound is due

to the discretization error $\Delta_m(N_b)$, and it decreases monotonically as N_b increases. From this point onwards, we ignore integrality of N_b and n_b to simplify analysis, since it will not affect the subsequent rate analysis as our focus is on asymptotic rate of decay of error bounds, when N_b and $n_b \to \infty$. Since N_b and n_b are constrained by the inequality in (III.21), for any N_1 and n_1 that satisfy (III.21) with strict inequality, there must exist a $N_2 \geq N_1$ and a $n_2 \geq n_1$ that satisfy (III.21) with equality, i.e.,

$$\sigma N_b^{\nu} n_b = b, \tag{III.22}$$

which produces a smaller error bound. Thus, we use (III.22) instead of (III.21) for subsequent analysis.

Let $\{N_b\}_{b\in\mathbb{N}}$ and $\{n_b\}_{b\in\mathbb{N}}$ be sequences that satisfy (III.22) for all $b \in \mathbb{N}$. We define $\{(N_b, n_b)\}_{b\in\mathbb{N}}$ as a *candidate selection*. Suppose that a particular algorithm has error bound $e_b, b \in \mathbb{N}$. Obviously, there are many candidate selections that make $\{e_b\}_{b\in\mathbb{N}}$ converge to zero. However, some candidate selections result in faster rates than others, and we want to find these selections. We note that the topic of determining algorithm parameter values to optimize algorithm efficiency has been addressed in the area of simulation optimization (Pasupathy, 2010; Lee & Glynn, 2003).

We first consider the rate of decay of error bound e_b for an *ideal* algorithm map, which solves (SMX_{N_b}) exactly in one iteration for any $N_b \in \mathbb{N}$.

2. Ideal Algorithm Map

Suppose that Assumptions III.3 and III.5 hold. Suppose also that a computational budget $b \in \mathbb{N}$ is allocated to Algorithm III.1 with an ideal algorithm map to solve (SMX_{N_b}). Since (SMX_{N_b}) is solved exactly in one iteration, $\psi_{N_b}(x_1^{N_b}) - \psi^*_{N_b} = 0$. Based on Proposition III.1(iii) and (III.10),

$$\psi(x_1^{N_b}) - \psi^* \leq \psi_{N_b}(x_1^{N_b}) + \Delta_m(N_b) - \psi^*_{N_b} = \Delta_m(N_b) = L' \frac{\sqrt{m}}{N_b^{1/m}}, \tag{III.23}$$

where L' is as in Assumption III.3. We define

$$e_b^{\text{ideal}} \triangleq \frac{L'\sqrt{m}}{N_b^{1/m}}. \tag{III.24}$$

Theorem III.6. *Suppose that Assumptions III.3 and III.5 hold. Suppose also that a computational budget $b \in \mathbb{N}$ is allocated to Algorithm III.1 with an ideal algorithm map to solve* (SMX_{N_b}). *Then the error bound*

$$e_b^{\text{ideal}} = \frac{L'\sqrt{m}\sigma^{1/m\nu}}{b^{1/m\nu}} \tag{III.25}$$

for all $b \in \mathbb{N}$, where L' is as in Assumption III.3, and σ and ν are as in Assumption III.5.

Proof. Since an ideal algorithm map is used, $n_b = 1$ for all $b \in \mathbb{N}$, and from (III.22), $N_b = (b/\sigma)^{1/\nu}$. The conclusion follows by substituting N_b into (III.24). □

The result above states that e_b^{ideal} decays at an asymptotic sublinear rate of $b^{-1/m\nu}$ as $b \to \infty$. Since the ideal algorithm map solves the discretized problems exactly (in one iteration), the rate-of-decay result for e_b^{ideal} determines the rate at which the error between the function values at the solutions of the discretized problems and the function value at the solution of the semi-infinite problem decays, as the level of discretization increases. Similarly, the rate-of-convergence results in Still (2001) and Shapiro (2009) determine the rate at which the error between the solutions of the discretized problems and the solution of the semi-infinite problem decays, as the level of discretization increases. Thus the rate-of-decay result for e_b^{ideal} is related to the rate-of-convergence results in Still (2001) and Shapiro (2009).

3. Adaptive Discretization Algorithm

The preceding result for fixed discretization can be generalized for a potentially more efficient adaptive discretization algorithm as follows. For the following adaptive discretization algorithm, we adopt a different notation (from the fixed discretization algorithm) for the iterates, specifically, we denote the j^{th} iterate at the i^{th} stage by $x_{i,j}$.

Algorithm III.2. Adaptive Discretization Algorithm

Data: $x_0 \in \mathbb{R}^d$.

Parameters: Number of stages $s \in \mathbb{N}$, discretization parameters $\{N_i\}_{i=1}^s, N_i \in \mathbb{N}$, number of iterations in the stages $\{n_i\}_{i=1}^s, n_i \in \mathbb{N}$, and parameters required for the algorithm map.

Step 1. Set $i = 1$.

Step 2. If $i > 1$, warm-start from the last iterate of the previous stage by setting $x_{i,1} = x_{i-1,n_{i-1}}$. Else, set $x_{i,1} = x_0$.

Step 3. Generate a sequence $\{x_{i,j}\}_{j=1}^{n_i}$ by applying a finite minimax algorithm map to (SMX_{N_i}).

Step 4. If $i < s$, replace i by $i + 1$, and go to Step 2. Else, end. □

Suppose that a computational budget $b \in \mathbb{N}$ is allocated to Algorithm III.2 with an algorithm map with an arbitrary rate of convergence to solve (SMX). Suppose also that Assumptions III.3 and III.5 hold.

Based on Proposition III.1(iii) and (III.10),

$$\psi(x_{s,n_s}) - \psi^* \leq \psi_{N_s}(x_{s,n_s}) + \Delta_m(N_s) - \psi^*_{N_s}. \tag{III.26}$$

We define

$$e_b^{\text{adaptive}} \triangleq \psi_{N_s}(x_{s,n_s}) + \Delta_m(N_s) - \psi^*_{N_s}. \tag{III.27}$$

Proposition III.7. *The error bound*

$$e_b^{\text{adaptive}} \geq \frac{L'\sqrt{m}\sigma^{1/m\nu}}{b^{1/m\nu}} \tag{III.28}$$

for all $b \in \mathbb{N}$, where L' is as in Assumption III.3, and σ and ν are as in Assumption III.5.

Proof. The parameters for Algorithm III.2, $s \in \mathbb{N}$, $\{N_i\}_{i=1}^s, N_i \in \mathbb{N}$, and $\{n_i\}_{i=1}^s, n_i \in \mathbb{N}$ satisfy

$$\sigma\left(N_1^\nu n_1 + N_2^\nu n_2 + ... + N_s^\nu n_s\right) = b. \tag{III.29}$$

This implies that $\sigma N_s^\nu \leq b$, and thus,

$$e_b^{\text{adaptive}} = \psi_{N_s}(x_{s,n_s}) + \Delta_m(N_s) - \psi_{N_s}^* \geq \Delta_m(N_s) \geq \frac{L'\sqrt{m}\sigma^{1/m\nu}}{b^{1/m\nu}}, \qquad \text{(III.30)}$$

based on (III.11) and the assumption that $\Delta_m(\cdot)$ is a monotonically decreasing function. □

The result above indicates that the e_b^{adaptive} for Algorithm III.2 with any algorithm map of any convergence rate, asymptotically decays with a rate no faster than $b^{-1/m\nu}$, as $b \to \infty$. In the following subsections, we show that this optimal rate of $b^{-1/m\nu}$ can be achieved using the fixed-discretization Algorithm III.1 with certain algorithm maps.

We say that an algorithm map converges uniformly when applied to (SMX_N), $N \in \mathbb{N}$, if the respective constants c and n_1 in Section I.D.2 do not depend on N.

4. Quadratically Convergent Algorithm Map

We obtain an error bound for Algorithm III.1 with a uniform quadratically convergent algorithm map in the next lemma. We refer to Section I.D for definitions of the various rates of convergence and uniform convergence.

Lemma III.8. *Suppose that Assumptions III.3 and III.4 hold. Suppose also that Algorithm III.1 with a uniform quadratically convergent algorithm map is used to solve* (SMX)*, i.e., there exist* $n_1 \in \mathbb{N}_0, n_1 < \infty$*, and* $c_1 \in (0, \infty)$ *such that*

$$\frac{\psi_N(x_{n+1}^N) - \psi_N^*}{[\psi_N(x_n^N) - \psi_N^*]^2} \leq c_1, \qquad \text{(III.31)}$$

for all $n \geq n_1$*. Then there exist* c*,* $\kappa < \infty$ *such that for all* $n \geq n_1$ *and* $N \in \mathbb{N}$*,*

$$\psi(x_n^N) - \psi^* \leq c^{2^n}\kappa + \Delta_m(N). \qquad \text{(III.32)}$$

Proof. Based on Proposition III.1(iii), (III.10), and (III.31),

$$\begin{aligned} \psi(x_n^N) - \psi^* &\leq \psi_N(x_n^N) + \Delta_m(N) - \psi_N^* \\ &\leq c_1^{2^{n-n_1}-1}[\psi_N(x_{n_1}^N) - \psi_N^*]^{2^{n-n_1}} + \Delta_m(N). \qquad \text{(III.33)} \end{aligned}$$

From (III.10), $-\psi_N^* \leq -\psi(x_N^*) + \Delta_m(N)$. Based on Assumption III.3, $\Delta_m(N)$ is a monotonically decreasing function, thus $\Delta_m(N) \leq \Delta_m(N_1)$ for all $N \in \mathbb{N}, N \geq N_1$. Since $\psi(x_N^*) \geq \psi(x^*)$ by definition,

$$\begin{aligned} & c_1^{2^{n-n_1-1}}[\psi_N(x_{n_1}^N) - \psi_N^*]^{2^{n-n_1}} \\ \leq \; & \left[c_1\left(\psi(x_{n_1}^N) - \psi^* + \Delta_m(N_1)\right)\right]^{2^n} \frac{\left[c_1\left(\psi(x_{n_1}^N) - \psi^* + \Delta_m(N_1)\right)\right]^{2^{-n_1}}}{c_1} \end{aligned} \tag{III.34}$$

where N_1 is as in Assumption III.3. Above we use the fact that for uniform convergence, n_1 is independent of N.

Under Assumption III.4, $x_{n_1}^N$ is bounded for any $n_1 \in \mathbb{N}$ and $N \in \mathbb{N}, N \geq N_1$. Since $\psi(\cdot)$ is continuous, $\psi(x_{n_1}^N)$ is bounded for any $n_1 \in \mathbb{N}$ and $N \in \mathbb{N}, N \geq N_1$. Based on Assumption III.4, ψ^* is finite. As $N_1 \in \mathbb{N}$, $\Delta_m(N_1) < \infty$ based on Assumption III.3 and (III.11). Finally, c_1 and n_1 are independent of N based on the assumption of uniform convergence, thus c and $\kappa < \infty$. □

From (III.32), we define the error bound for Algorithm III.1 with a quadratically convergent algorithm map as

$$e_b^{\text{quad}} \triangleq c^{2^{n_b}}\kappa + \Delta_m(N_b). \tag{III.35}$$

The next result states that if we choose the candidate selections in a certain way, then a fixed discretization algorithm with a quadratically convergent algorithm map can achieve the same optimal asymptotic rate of decay of error bound as Algorithm III.2.

We use $\log(\cdot)$ to denote the natural logarithm.

Theorem III.9. *Suppose that Assumptions III.3, III.4, and III.5 hold. Suppose also that a computational budget $b \in \mathbb{N}$ is allocated to Algorithm III.1 with a uniform quadratically convergent algorithm map with rate of convergence given by (III.31) to solve* (SMX)*. If*

$$N_b = \left(\frac{b \log 2}{\sigma[\log\log(b/\sigma) - \log(-m\nu \log c)]}\right)^{1/\nu} \tag{III.36}$$

and

$$n_b = \frac{\log\log(b/\sigma) - \log(-m\nu \log c)}{\log 2} \tag{III.37}$$

for all $b \in \mathbb{N}$, then (III.22) is satisfied and

$$\lim_{b\to\infty} \frac{\log e_b^{\text{quad}}}{\log b} = -\frac{1}{m\nu}, \tag{III.38}$$

where $m \in \mathbb{N}$ is the uncertainty dimension and ν is as defined in Assumption III.5.

Proof. From (III.11), (III.22), and (III.32),

$$\begin{aligned}
\log e_b^{\text{quad}} &= \log\left(\exp\left[\log\kappa + 2^{b/(\sigma N_b^\nu)}\log c\right] + \exp\left[\log L'\sqrt{m} - \log(N_b^{1/m})\right]\right) \\
&= \log\Bigg(\exp\left[\log\kappa + 2^{[\log\log(b/\sigma)-\log(-m\nu\log c)]/\log 2}\log c\right] \\
&+ \exp\left[\log L'\sqrt{m} - \frac{1}{m\nu}\log\left(\frac{b\log 2}{\sigma[\log\log(b/\sigma) - \log(-m\nu\log c)]}\right)\right]\Bigg) \\
&= \log\Bigg(\exp\left[\log\kappa + (\log c)\exp[\log\log(b/\sigma) - \log(-m\nu\log c)]\right] \\
&+ \exp\left[\log L'\sqrt{m} - \frac{1}{m\nu}\log\left(\frac{b\log 2}{\sigma[\log\log(b/\sigma) - \log(-m\nu\log c)]}\right)\right]\Bigg) \\
&= \log\left(\exp\left[\log\kappa + (\log c)\frac{\log\frac{b}{\sigma}}{-m\nu\log c}\right]\right. \\
&+ \left.\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma) - \log(-m\nu\log c)]}{b\log 2}\right)^{\frac{1}{m\nu}}\right) \\
&= \log\left(\kappa\left(\frac{b}{\sigma}\right)^{-\frac{1}{m\nu}} + \left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma) - \log(-m\nu\log c)]}{b\log 2}\right)^{\frac{1}{m\nu}}\right) \\
&= \log\Bigg(\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma) - \log(-m\nu\log c)]}{b\log 2}\right)^{\frac{1}{m\nu}} \\
&\times \left\{\frac{\kappa\left(\frac{b}{\sigma}\right)^{-\frac{1}{m\nu}}}{\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma)-\log(-m\nu\log c)]}{b\log 2}\right)^{\frac{1}{m\nu}}} + 1\right\}\Bigg),
\end{aligned} \tag{III.39}$$

where we use the fact that $2^{x/\log 2} = \exp(x)$ for $x \in \mathbb{R}$. Simplifying the second term within the $\log(\cdot)$ function, we obtain that

$$\frac{\kappa\left(\frac{b}{\sigma}\right)^{-\frac{1}{m\nu}}}{\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma)-\log(-m\nu\log c)]}{b\log 2}\right)^{\frac{1}{m\nu}}} = \frac{\kappa\left(\frac{1}{\sigma}\right)^{-\frac{1}{m\nu}}}{\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma)-\log(-m\nu\log c)]}{\log 2}\right)^{\frac{1}{m\nu}}}. \quad \text{(III.40)}$$

Since

$$\lim_{b\to\infty} \frac{\kappa\left(\frac{1}{\sigma}\right)^{-\frac{1}{m\nu}}}{\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma)-\log(-m\nu\log c)]}{\log 2}\right)^{\frac{1}{m\nu}}} = 0, \quad \text{(III.41)}$$

then by continuity of the $\log(\cdot)$ function,

$$\lim_{b\to\infty} \log\left(\frac{\kappa\left(\frac{b}{\sigma}\right)^{-\frac{1}{m\nu}}}{\left(\frac{L'\sqrt{m}\sigma[\log\log(b/\sigma)-\log(-m\nu\log c)]}{b\log 2}\right)^{\frac{1}{m\nu}}} + 1\right) = 0. \quad \text{(III.42)}$$

Therefore, continuing from (III.39),

$$\begin{aligned}\lim_{b\to\infty} \frac{\log e_b^{\text{quad}}}{\log b} &= \lim_{b\to\infty} \frac{1}{m\nu}\left(\frac{\log L'\sqrt{m}\sigma}{\log b} + \frac{\log[\log\log(b/\sigma) - \log(-m\nu\log c)]}{\log b}\right. \\ &\quad \left. - \frac{\log b}{\log b} - \frac{\log\log 2}{\log b}\right) = -\frac{1}{m\nu}. \quad \text{(III.43)}\end{aligned}$$

This completes the proof. □

Roughly, what Theorem III.9 says is, if you make certain choices for the discretization, by picking N_b and n_b as in (III.36) and (III.37), respectively, for large b, if b increases by a factor $b_1 \in (1,\infty)$, then $e_b^{\text{quadratic}}$ decreases by a factor $b_1^{-\frac{1}{m\nu}}$.

5. Linearly Convergent Algorithm Map

We next obtain an error bound for Algorithm III.1 with a uniform linearly convergent algorithm map.

Lemma III.10. *Suppose that Assumptions III.3 and III.4 hold. Suppose also that Algorithm III.1 with a uniform linearly convergent algorithm map is used to solve* (SMX)*, i.e., there exist* $n_1 \in \mathbb{N}_0$*,* $n_1 < \infty$*,* $N_1 \in \mathbb{N}$*, and* $c \in (0,1)$ *such that*

$$\frac{\psi_N(x_{n+1}^N) - \psi_N^*}{\psi_N(x_n^N) - \psi_N^*} \le c, \quad \text{(III.44)}$$

for all $n \geq n_1$ and $N \geq N_1$. Then there exists a $\kappa < \infty$ such that for all $n \geq n_1$ and $N \geq N_1$,

$$\psi(x_n^N) - \psi^* \leq c^n \kappa + \Delta_m(N). \tag{III.45}$$

Proof. Based on Proposition III.1(iii), (III.10), (III.44), and using similar arguments as the proof for Lemma III.8,

$$\begin{aligned} \psi(x_n^N) - \psi^* &\leq \psi_N(x_n^N) + \Delta_m(N) - \psi_N^* \\ &\leq c^{n-n_1}[\psi_N(x_{n_1}^N) - \psi_N^*] + \Delta_m(N) \\ &\leq c^n(c^{-n_1}[\psi(x_{n_1}^N) - \psi^* + \Delta_m(N_1)]) + \Delta_m(N), \end{aligned} \tag{III.46}$$

where N_1 is as in Assumption III.3. The remaining part of the proof follows the same arguments as the proof for Lemma III.8. □

From (III.45), we define the error bound for Algorithm III.1 with a linearly convergent algorithm map as

$$e_b^{\text{linear}} \stackrel{\triangle}{=} c^{n_b}\kappa + \Delta_m(N_b). \tag{III.47}$$

The next result states that a fixed discretization algorithm with a linearly convergent algorithm map can achieve the same asymptotic rate of decay of error bound as Algorithm III.2.

Theorem III.11. *Suppose that Assumptions III.3, III.4, and III.5 hold. Suppose also that a computational budget $b \in \mathbb{N}$ is allocated to Algorithm III.1 with a uniform linearly convergent algorithm map with rate of convergence given by (III.44) to solve* (SMX)*. If*

$$N_b = \left(-\frac{mb\nu \log c}{\sigma \log b}\right)^{1/\nu} \tag{III.48}$$

and

$$n_b = -\frac{b \log b}{mb\nu \log c} \tag{III.49}$$

for all $b \in \mathbb{N}$, then (III.22) is satisfied and

$$\lim_{b\to\infty} \frac{\log e_b^{\text{linear}}}{\log b} = -\frac{1}{m\nu}, \tag{III.50}$$

where $m \in \mathbb{N}$ is the uncertainty dimension and ν is as defined in Assumption III.5.

Proof. From (III.11), (III.22), and (III.45),

$$\begin{aligned}
\log e_b^{\text{linear}} &= \log\left(\exp\left[\frac{b\log c}{\sigma N_b^\nu} + \log\kappa\right] + \exp\left[\log L'\sqrt{m} - \log(N_b^{1/m})\right]\right) \\
&= \log\left(\exp\left[\frac{-\sigma b\log b\log c}{\sigma m b\nu\log c} + \log\kappa\right]\right. \\
&+ \left.\exp\left[\log L'\sqrt{m} - \log\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)\right]\right) \\
&= \log\left(\exp\left[\log L'\sqrt{m} - \log\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)\right]\times\right. \\
&\left.\left\{\frac{\exp\left[\frac{-\log b}{m\nu} + \log\kappa\right]}{\exp\left[\log L'\sqrt{m} - \log\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)\right]} + 1\right\}\right). \qquad \text{(III.51)}
\end{aligned}$$

Simplifying the second term within the outermost $\log(\cdot)$ function,

$$\begin{aligned}
\frac{\exp\left[\frac{-\log b}{m\nu} + \log\kappa\right]}{\exp\left[\log L'\sqrt{m} - \log\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)\right]} &= \frac{\kappa\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)}{b^{1/m\nu}L'\sqrt{m}} \\
&= \frac{\kappa\left(\left[\frac{-m\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)}{L'\sqrt{m}}. \qquad \text{(III.52)}
\end{aligned}$$

Since

$$\lim_{b\to\infty}\frac{\kappa\left(\left[\frac{-m\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)}{L'\sqrt{m}} = 0, \qquad \text{(III.53)}$$

then by continuity of the $\log(\cdot)$ function,

$$\lim_{b\to\infty}\log\left\{\frac{\exp\left[\frac{-\log b}{m\nu} + \log\kappa\right]}{\exp\left[\log L'\sqrt{m} - \log\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)\right]} + 1\right\} = 0. \qquad \text{(III.54)}$$

Therefore, from (III.51),

$$\begin{aligned}\lim_{b\to\infty}\frac{\log e_b^{\text{linear}}}{\log b} &= \lim_{b\to\infty}\frac{\log L'\sqrt{m}}{\log b} - \frac{\log\left(\left[\frac{-mb\nu\log c}{\sigma\log b}\right]^{1/m\nu}\right)}{\log b} \\ &= \lim_{b\to\infty} -\frac{1}{m\nu}\left(\frac{\log(-m\nu\log c)+\log b-\log\sigma-\log\log b}{\log b}\right) \\ &= -\frac{1}{m\nu}. \end{aligned} \tag{III.55}$$

This completes the proof. □

6. Sublinearly Convergent Algorithm Map

Since both the quadratically and linearly convergent algorithm maps obtain the ideal rate of decay of error bound of $b^{-1/m\nu}$, one may think that the rate of the algorithm map does not matter. But that is not the case, as the following counter example shows. We next obtain an error bound for Algorithm III.1 with a uniform sublinearly convergent algorithm map. We define the initial error $e_0 \stackrel{\triangle}{=} \psi(x_0) - \psi^*$.

Lemma III.12. *Suppose that Assumption III.3 holds. Suppose also that Algorithm III.1 with a uniform sublinearly convergent algorithm map is used to solve* (SMX)*, i.e., there exist* $N_1 \in \mathbb{N}$ *and* $a > 1$ *such that*

$$\frac{\psi_N(x_{n+1}^N) - \psi_N^*}{\psi_N(x_n^N) - \psi_N^*} \leq 1 - \frac{1}{n+a} \tag{III.56}$$

for all $n \in \mathbb{N}$*,* $N \in \mathbb{N}$*, and* $N \geq N_1$*. Then for any* $n \in \mathbb{N}$*,* $N \in \mathbb{N}, N \geq N_1$*,*

$$\psi\left(x_n^N\right) - \psi^* \leq \frac{a-1}{n-1+a}e_0 + 2\Delta_m(N). \tag{III.57}$$

Proof: Based on Proposition III.1, (III.10), and (III.56),

$$\begin{aligned}
&\psi\left(x_n^N\right) - \psi^* \\
&\leq \psi_N\left(x_n^N\right) + \Delta_m(N) - \psi_N^* \\
&\leq \left(1 - \frac{1}{a}\right)\left(1 - \frac{1}{1+a}\right) ... \left(1 - \frac{1}{n-1+a}\right)[\psi_N(x_0) - \psi_N^*] + \Delta_m(N) \\
&= \left(\frac{a-1}{a}\right)\left(\frac{a}{1+a}\right) ... \left(\frac{n-2+a}{n-1+a}\right)[\psi_N(x_0) - \psi_N^*] + \Delta_m(N) \\
&\leq \frac{a-1}{n-1+a}[\psi(x_0) - \psi^* + \Delta_m(N)] + \Delta_m(N) \\
&\leq \frac{a-1}{n-1+a}e_0 + 2\Delta_m(N). \qquad \text{(III.58)}
\end{aligned}$$

This completes the proof. □

From (III.57), we define the error bound for Algorithm III.1 with a sublinearly convergent algorithm map as

$$e_b^{\text{sub}} \triangleq \frac{a-1}{n_b - 1 + a}e_0 + 2\Delta_m(N_b). \qquad \text{(III.59)}$$

The next result states that a fixed discretization algorithm with a sublinearly convergent algorithm map is unable to achieve the same asymptotic rate of decay of error bound as Algorithm III.2.

Theorem III.13. *Suppose that Assumptions III.3, III.4, and III.5 hold. Suppose also that a computational budget $b \in \mathbb{N}$ is allocated to Algorithm III.1 with a uniform sublinearly convergent algorithm map with rate of convergence given by (III.56) to solve* (SMX)*. Then for all possible sequences of $\{(N_b, n_b)\}_{b\in\mathbb{N}}$,*

$$\liminf_{b\to\infty} \frac{\log e_b^{\text{sub}}}{\log b} > -\frac{1}{m\nu}, \qquad \text{(III.60)}$$

where $m \in \mathbb{N}$ is the uncertainty dimension and ν is as defined in Assumption III.5.

Proof. From (III.11), (III.22), and (III.57),

$$\begin{aligned}
\log e_b^{\text{sub}} &= \log\Bigg(\exp\left[\log(a-1)+\log e_0-\log(n_b-1+a)\right] \\
&+ \exp\left[\log 2L'\sqrt{m}-\frac{\log b}{m\nu}+\frac{\log \sigma n_b}{m\nu}\right]\Bigg) \\
&\geq \log\Bigg(\max\Bigg\{\exp\left[\log(a-1)+\log e_0-\log(n_b-1+a)\right], \\
&\qquad \exp\left[\log 2L'\sqrt{m}-\frac{\log b}{m\nu}+\frac{\log \sigma n_b}{m\nu}\right]\Bigg\}\Bigg) \\
&= \max\Bigg\{\log\left(\exp\left[\log(a-1)+\log e_0-\log(n_b-1+a)\right]\right), \\
&\qquad \log\left(\exp\left[\log 2L'\sqrt{m}-\frac{\log b}{m\nu}+\frac{\log \sigma n_b}{m\nu}\right]\right)\Bigg\} \\
&= \max\Bigg\{\log(a-1)+\log e_0-\log(n_b-1+a), \\
&\qquad \log 2L'\sqrt{m}-\frac{\log b}{m\nu}+\frac{\log \sigma n_b}{m\nu}\Bigg\}. \qquad \text{(III.61)}
\end{aligned}$$

Hence, for any $b \geq 2$,

$$\begin{aligned}
\frac{\log e_b^{\text{sub}}}{\log b} &\geq \max\Bigg\{\frac{\log(a-1)}{\log b}+\frac{\log e_0}{\log b}-\frac{\log(n_b-1+a)}{\log b}, \\
&\qquad \frac{\log 2L'\sqrt{m}}{\log b}-\frac{\log b}{m\nu\log b}+\frac{\log \sigma n_b}{m\nu\log b}\Bigg\}. \qquad \text{(III.62)}
\end{aligned}$$

For the sake of contradiction, we assume that there exists a sequence $\{N_b, n_b\}_{b\geq 2}$ where

$$\liminf_{b\to\infty}\frac{\log e_b^{\text{sub}}}{\log b} \leq -\frac{1}{m\nu}. \qquad \text{(III.63)}$$

This implies that for every $\epsilon > 0$, there exists an infinite subsequence $\mathcal{B} \subset \mathbb{N}$, a $b_1 \in \mathcal{B}, b_1 \geq 2$ such that

$$\frac{\log e_b^{\text{sub}}}{\log b} \leq -\frac{1}{m\nu}+\epsilon, \qquad \text{(III.64)}$$

for all $b \in \mathcal{B}, b \geq b_1$. From (III.62) and (III.64),

$$\frac{\log(a-1)}{\log b}+\frac{\log e_0}{\log b}-\frac{\log(n_b-1+a)}{\log b} \leq -\frac{1}{m\nu}+\epsilon, \qquad \text{(III.65)}$$

and

$$\frac{\log 2L'\sqrt{m}}{\log b} - \frac{1}{m\nu} + \frac{\log \sigma n_b}{m\nu \log b} \leq -\frac{1}{m\nu} + \epsilon, \tag{III.66}$$

for all $b \in \mathcal{B}, b \geq b_1$. From (III.65) and (III.66), there exists a $b_2 \in \mathcal{B}, b_2 \geq b_1$ such that

$$\frac{\log n_b}{\log b} \geq \frac{1}{m\nu} - 2\epsilon, \tag{III.67}$$

and

$$\frac{\log n_b}{\log b} \leq 2m\nu\epsilon, \tag{III.68}$$

for all $b \in \mathcal{B}, b \geq b_2$. Since ϵ is arbitrary, (III.67) contradicts (III.68) for sufficiently small ϵ, and the conclusion follows. □

7. Smoothing Algorithm Map

In this subsection, we analyze the rate of decay of error bound for Algorithm III.1 using smoothing algorithms as algorithm maps. We first repeat some of the known results on the exponential smoothing technique from Section II.B, based on the assumptions and notation in this chapter.

For any $p > 0$ and $N \in \mathbb{N}$, we consider a smooth approximating problem to the generally non-differentiable (SMX_N),

$$(\mathrm{SMX}_{Np}) \quad \min_{x \in \mathbb{R}^d} \psi_{Np}(x), \tag{III.69}$$

where

$$\psi_{Np}(x) \triangleq \frac{1}{p} \log \left(\sum_{y \in Y_N} \exp\left(p\phi(x,y)\right) \right) \tag{III.70}$$

$$= \psi_N(x) + \frac{1}{p} \log \left(\sum_{y \in Y_N} \exp\left(p(\phi(x,y) - \psi_N(x))\right) \right) \tag{III.71}$$

is the exponential penalty function.

The parameter $p > 0$ is the smoothing precision parameter, where a larger p implies higher precision. With the obvious notational changes, we have a similar result

on the bounds for $\psi_{Np}(x) - \psi_N(x)$ and the differentiability of $\psi_{Np}(\cdot)$ as Proposition II.1.

Assumption III.14. *The function $\phi(\cdot,\cdot)$ is twice continuously differentiable on $\mathbb{R}^d \times Y$.* □

Lemma III.15. *Suppose that Assumption III.14 holds. Then for every bounded set $S \subset \mathbb{R}^d$, there exists an $L < \infty$ such that*

$$\langle z, \nabla^2 \psi_{Np}(x) z \rangle \leq pL\|z\|^2, \tag{III.72}$$

for all $x \in S, z \in \mathbb{R}^d$, $N \in \mathbb{N}$, and $p \geq 1$.

Proof. The proof follows similar arguments as the proof for Lemma II.7 on p. 20. □

When they exist, we denote the optimal value of (SMX_{Np}) by ψ^*_{Np} for any $N \in \mathbb{N}$ and $p > 0$, and the optimal solution of (SMX_{Np}) by x^*_{Np}. We denote the n^{th} iterate of a sequence generated by an algorithm map when applied to (SMX_{Np}) by x_n^{Np}.

Lemma III.16. *Suppose that Assumption III.4 holds. For any $x, z \in \mathbb{R}^d, N \in \mathbb{N}$, and $p > 0$,*

$$a\|z\|^2 \leq \left\langle z, \nabla^2 \psi_{Np}(x) z \right\rangle, \tag{III.73}$$

where a satisfies the inequality in Assumption III.4.

Proof. The proof follows the same arguments as the proof for Lemma II.5 on p. 19. □

The Armijo Gradient Method is referenced in the following proposition. The Armijo Gradient Method uses the steepest descent search direction and the Armijo stepsize rule to solve an unconstrained problem; see for example Algorithm 1.3.3 of Polak (1997).

Proposition III.17. *Suppose that Assumption III.4 holds, $N \in \mathbb{N}$, and $p \geq 1$. Then the rate of convergence for the Armijo Gradient Method to solve (SMX_{Np}) is linear*

with coefficient $1-k/p$*, for some* $k \in (0,1)$*. That is, for any sequence* $\{x_n^{Np}\}_{n=0}^{\infty} \subset \mathbb{R}^d$ *generated by the Armijo Gradient Method when applied to* (SMX_{Np})*, there exists a* $k \in (0,1)$ *such that*

$$\psi_{Np}(x_{n+1}^{Np}) - \psi_{Np}^* \leq \left(1 - \frac{k}{p}\right) [\psi_{Np}(x_n^{Np}) - \psi_{Np}^*)] \quad \text{for all } n \in \mathbb{N}_0. \tag{III.74}$$

Proof. The proof follows the same arguments as the proof for Proposition II.8 on p. 22. □

Lemma III.18. *Suppose that Assumptions III.3 and III.4 hold. If the Armijo Gradient method is applied on* (SMX_{Np})*, where* $p \geq 1$*,* $N \in \mathbb{N}, N \geq N_1$*, and* N_1 *is as defined in Assumption III.3. Then for any* $n \in \mathbb{N}$*,*

$$\psi(x_n^{Np}) - \psi^* \leq \left(1 - \frac{k}{p}\right)^n e_0 + 2\Delta_m(N) + \frac{2\log N}{p}, \tag{III.75}$$

where $k \in (0,1)$ *is the constant in Proposition III.17.*

Proof. Since $\phi(\cdot, y)$ is twice continuously differentiable for all $y \in Y$, with an equivalent result for $\psi_{Np}(\cdot)$ as Proposition II.1, $\psi_{Np}(\cdot)$ is continuous and

$$0 \leq \psi_{Np}(x) - \psi_N(x) \leq \frac{\log N}{p} \tag{III.76}$$

for all $N \in \mathbb{N}$, $p > 0$, and $x \in \mathbb{R}^d$. Based on Proposition III.17,

$$\begin{aligned}
& \psi(x_n^{Np}) - \psi^* \\
\leq\ & \psi_N(x_n^{Np}) + \Delta_m(N) - \psi_N(x^*) \\
\leq\ & \psi_{Np}(x_n^{Np}) + \Delta_m(N) - \psi_{Np}(x^*) + (\log N)/p \\
\leq\ & \psi_{Np}(x_n^{Np}) - \psi_{Np}^* + \Delta_m(N) + (\log N)/p \\
\leq\ & (1-(k/p))^n \left[\psi_{Np}(x_0) - \psi_{Np}^*\right] + \Delta_m(N) + (\log N)/p \\
\leq\ & (1-(k/p))^n \left[\psi_N(x_0) + (\log N)/p - \psi_N(x_{Np}^*)\right] + \Delta_m(N) + (\log N)/p \\
\leq\ & (1-(k/p))^n \left[\psi(x_0) + (\log N)/p - \psi(x_{Np}^*) + \Delta_m(N)\right] + \Delta_m(N) + (\log N)/p \\
\leq\ & (1-(k/p))^n [\psi(x_0) - \psi(x^*)] + 2\Delta_m(N) + (2\log N)/p.
\end{aligned} \tag{III.77}$$

This completes the proof. □

From (III.75), we define the error bound for Algorithm III.1 with a smoothing algorithm map as

$$e_b^{\text{smooth}} \triangleq \left(1 - \frac{k}{p_b}\right)^{n_b} e_0 + 2\Delta_m(N_b) + \frac{2\log N_b}{p_b}. \tag{III.78}$$

The next result states that a fixed discretization algorithm with a smoothing algorithm map is unable to achieve the same asymptotic rate of decay of error bound as Algorithm III.2.

Theorem III.19. *Suppose that Assumptions III.3, III.4, and III.5 hold. Suppose also that a computational budget $b \in \mathbb{N}$ is expended by running $n_b \in \mathbb{N}$ iterations of the Armijo Gradient method on $(\text{SMX}_{N_b p_b})$, with discretization parameter $N_b \in \mathbb{N}, N_b \geq N_1$ as defined in Assumption III.3, and smoothing parameter $p_b \geq 1$. Then for all possible sequences of $\{N_b, n_b, p_b\}_{b\in\mathbb{N}}$ satisfying (III.22),*

$$\liminf_{b\to\infty} \frac{\log e_b^{\text{smooth}}}{\log b} > -\frac{1}{m\nu}, \tag{III.79}$$

where $m \in \mathbb{N}$ is the uncertainty dimension and ν is as defined in Assumption III.5.

Proof. From (III.11), and (III.75),

$$\begin{aligned}
\log e_b^{\text{smooth}} &= \log\left(\exp\left[\frac{b}{\sigma N_b^\nu}\log\left(1-\frac{k}{p_b}\right)+\log e_0\right]+\exp\left[\log\frac{2L'\sqrt{m}}{N_b^{1/m}}\right]\right. \\
&+ \left.\exp\left[\log\frac{2\log N_b}{p_b}\right]\right) \\
&\geq \log\left(\max\left\{\exp\left[\frac{b}{\sigma N_b^\nu}\log\left(1-\frac{k}{p_b}\right)+\log e_0\right],\exp\left[\log\frac{2L'\sqrt{m}}{N_b^{1/m}}\right],\right.\right. \\
&\quad \left.\left.\exp\left[\log\frac{2\log N_b}{p_b}\right]\right\}\right) \\
&= \max\left\{\log\left(\exp\left[\frac{b}{\sigma N_b^\nu}\log\left(1-\frac{k}{p_b}\right)+\log e_0\right]\right),\right. \\
&\quad \left.\log\left(\exp\left[\log\frac{2L'\sqrt{m}}{N_b^{1/m}}\right]\right),\log\left(\exp\left[\log\frac{2\log N_b}{p_b}\right]\right)\right\} \\
&= \max\left\{\frac{b}{\sigma N_b^\nu}\log\left(1-\frac{k}{p_b}\right)+\log e_0,\log\frac{2L'\sqrt{m}}{N_b^{1/m}},\log\frac{2\log N_b}{p_b}\right\}. \qquad \text{(III.80)}
\end{aligned}$$

Hence, for any $b\in\mathbb{N}, b\geq 2$,

$$\frac{\log e_b^{\text{smooth}}}{\log b}\geq\max\left\{\frac{b}{\sigma N_b^\nu\log b}\log\left(1-\frac{k}{p_b}\right)+\frac{\log e_0}{\log b},\frac{\log\frac{2L'\sqrt{m}}{N_b^{1/m}}}{\log b},\frac{\log\frac{2\log N_b}{p_b}}{\log b}\right\} \qquad \text{(III.81)}$$

For the sake of contradiction, we assume that there exists a sequence $\{N_b,n_b,p_b\}_{b\in\mathbb{N}}$ where

$$\liminf_{b\to\infty}\frac{\log e_b^{\text{smooth}}}{\log b}\leq-\frac{1}{m\nu}. \qquad \text{(III.82)}$$

This implies that there exists an infinite subsequence $\mathcal{B}\subset\mathbb{N}$ such that

$$\lim_{b\to^{\mathcal{B}}\infty}\frac{\log e_b^{\text{smooth}}}{\log b}\leq-\frac{1}{m\nu}, \qquad \text{(III.83)}$$

which further implies that for any $\epsilon\in(0,\frac{\frac{1}{m\nu}}{1+m\nu})$, there exists a $b_1\in\mathcal{B}$ such that

$$\frac{\log e_b^{\text{smooth}}}{\log b}\leq-\frac{1}{m\nu}+\tfrac{1}{2}\epsilon, \qquad \text{(III.84)}$$

for all $b\in\mathcal{B}, b\geq b_1$. From (III.81) and (III.84), we have

$$\frac{b}{\sigma N_b^\nu\log b}\log\left(1-\frac{k}{p_b}\right)\leq-\frac{1}{m\nu}+\tfrac{1}{2}\epsilon, \qquad \text{(III.85)}$$

$$\frac{\log 2L'\sqrt{m}}{\log b} - \frac{\log N_b}{m \log b} \leq -\frac{1}{m\nu} + \tfrac{1}{2}\epsilon, \tag{III.86}$$

and

$$\frac{\log 2}{\log b} + \frac{\log \log N_b}{\log b} - \frac{\log p_b}{\log b} \leq -\frac{1}{m\nu} + \tfrac{1}{2}\epsilon, \tag{III.87}$$

for all $b \in \mathcal{B}, b \geq \max\{2, b_1\}$.

There exists a $b_2 \in \mathcal{B}, b_2 \geq b_1$ such that $(\log 2L'\sqrt{m})/\log b \geq -\frac{1}{2}\epsilon$ for all $b \geq b_2$. From (III.85)-(III.87), we get that

$$\frac{b}{\sigma N_b^{\nu} \log b} \log\left(1 - \frac{k}{p_b}\right) \leq -\frac{1}{m\nu} + \epsilon, \tag{III.88}$$

$$-\frac{\log N_b}{m \log b} \leq -\frac{1}{m\nu} + \epsilon, \tag{III.89}$$

and

$$\frac{\log \log N_b}{\log b} - \frac{\log p_b}{\log b} \leq -\frac{1}{m\nu} + \epsilon \tag{III.90}$$

for all $b \in \mathcal{B}, b \geq \max\{2, b_2\}$.

From (III.89), for all $b \in \mathcal{B}, b \geq \max\{2, b_2\}$,

$$N_b \geq b^{\frac{1}{\nu} - m\epsilon}. \tag{III.91}$$

From (III.90), for all $b \in \mathcal{B}, b \geq \max\{2, b_2\}$, $p_b \geq 1$,

$$\frac{\log\left(\frac{\log N_b}{p_b}\right)}{\log b} \leq -\frac{1}{m\nu} + \epsilon. \tag{III.92}$$

This implies that

$$p_b \geq \frac{\log N_b}{b^{-\frac{1}{m\nu} + \epsilon}}. \tag{III.93}$$

Next, we substitute $N_b \geq b^{\frac{1}{\nu} - m\epsilon}$ from (III.91) into (III.93), and we get

$$p_b \geq \frac{\log N_b}{b^{-\frac{1}{m\nu} + \epsilon}} \geq \frac{\log b^{\frac{1}{\nu} - m\epsilon}}{b^{-\frac{1}{m\nu} + \epsilon}} = \left(\frac{1}{\nu} - m\epsilon\right) b^{\frac{1}{m\nu} - \epsilon} \log b, \tag{III.94}$$

for all $b \in \mathcal{B}, b \geq \max\{2, b_2\}$. Using $N_b \geq b^{\frac{1}{\nu}-m\epsilon}$ from (III.91),

$$\frac{b}{\sigma N_b^\nu \log b} \leq \frac{b}{\sigma \left(b^{\frac{1}{\nu}-m\epsilon}\right)^\nu \log b} = \frac{b}{\sigma \left(b^{1-m\nu\epsilon}\right) \log b}, \tag{III.95}$$

for all $b \in \mathcal{B}, b \geq \max\{2, b_2\}$. Observe from (III.94) that if $\epsilon < \frac{\frac{1}{m\nu}}{1+m\nu}$, then $\frac{1}{m\nu} - \epsilon > 0$ and $p_b \to \infty$ as $b \to \infty$. Thus, there exists a $b_3 \in \mathcal{B}, b_3 \geq b_2$ such that $k/p_b \in [0, 1/2]$ for all $b \geq b_3$, and based on Lemma II.13,

$$\log\left(1 - \frac{k}{p_b}\right) \geq -\frac{2k}{p_b} \tag{III.96}$$

for all $b \in \mathcal{B}, b \geq b_3$.

Based on (III.94)-(III.96),

$$\begin{aligned} \frac{b}{\sigma N_b^\nu \log b} \log\left(1 - \frac{k}{p_b}\right) &\geq \frac{b}{\sigma \left(b^{1-m\nu\epsilon}\right) \log b} \frac{-2k}{\left(\frac{1}{\nu} - m\epsilon\right) b^{\frac{1}{m\nu}-\epsilon} \log b} \\ &= \frac{-2k}{\sigma \left(\frac{1}{\nu} - m\epsilon\right) b^{\frac{1}{m\nu}-m\nu\epsilon-\epsilon} \log^2 b}. \end{aligned} \tag{III.97}$$

As $\epsilon < \frac{\frac{1}{m\nu}}{1+m\nu}$, $\frac{1}{m\nu} - m\nu\epsilon - \epsilon > 0$ and $\frac{-2k}{\sigma\left(\frac{1}{\nu}-m\epsilon\right)b^{\frac{1}{m\nu}-m\nu\epsilon-\epsilon}\log^2 b} \to 0$ as $b \to \infty$. Thus, there exists a $b_4 \in \mathcal{B}$ such that (III.97) contradicts (III.88) for all $b \geq b_4$. This completes the proof. □

C. EFFICIENCY OF ϵ-SUBGRADIENT METHOD

Section III.B shows that discretization algorithms for solving (SMX) can obtain at best an asymptotic rate of decay of error bound of $b^{-1/m\nu}$ as $b \to \infty$, where m is the uncertainty dimension, ν is a parameter related to the work per iteration of the algorithm map, and b is the computational budget expended. Hence, discretization methods may perform poorly for (SMX) with large uncertainty dimension. In this section, we show that an ϵ-subgradient algorithm for (SMX), which relies on additional assumptions as compared to discretization algorithms, have more favorable rate of decay of error bound for moderate and large m.

This section starts with some definitions followed by a description of the ϵ-subgradient algorithm. We then we determine the rate of decay of an error bound

based on the ϵ-subgradient algorithm. Most of the background information on the ϵ-subgradient algorithm in this section are extracted from Bertsekas (2010).

We start by defining subgradients and subdifferentials.

Definition III.1. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. A vector $g \in \mathbb{R}^d$ is

(i) a *subgradient* of $f(\cdot)$ at a point $x \in \mathbb{R}^d$ if

$$f(z) \geq f(x) - (z - x)^T g \tag{III.98}$$

for all $z \in \mathbb{R}^d$,

(ii) an ϵ-*subgradient* of $f(\cdot)$ ($\epsilon > 0$) at a point $x \in \mathbb{R}^d$ if

$$f(z) \geq f(x) - (z - x)^T g - \epsilon \tag{III.99}$$

for all $z \in \mathbb{R}^d$.

(iii) The set of all subgradients of a convex function $f(\cdot)$ at $x \in \mathbb{R}^d$ is called the *subdifferential* of $f(\cdot)$ at $x \in \mathbb{R}^d$, which is denoted by $\partial f(x)$. □

We consider the following ϵ-subgradient algorithm.

Algorithm III.3. ϵ-Subgradient Algorithm

Data: $x_0 \in \mathbb{R}^d$.

Parameters: $\alpha > 0, \epsilon > 0$.

Step 1. Set $i = 0$.

Step 2. Compute $y_i \in Y$ such that

$$\phi(x_i, y_i) \geq \psi(x_i) - \epsilon. \tag{III.100}$$

Step 3. Determine the next iterate

$$x_{i+1} = x_i - \alpha \nabla_x \phi(x_i, y_i). \tag{III.101}$$

Step 4. Replace i by $i + 1$, and go to Step 2. □

The key step of Algorithm III.3 is Step 2, where we find a $y_i \in Y$ that has a value within ϵ of $\psi(x_i)$. Under the assumption that for all $y \in Y$, $\phi(\cdot, y)$ is convex for all $x \in \mathbb{R}^d$, the search direction $\nabla_x \phi(x_i, y_i)$ is an ϵ-subgradient of $\psi(\cdot)$ at x_i.

In Algorithm III.3, we use constant stepsize α. There are two other stepsize rules for subgradient algorithms, diminishing stepsize and dynamically-chosen stepsize. We refer to Bertsekas (2010, pp. 272-274) for a detailed discussion of the advantages and disadvantages of the three schemes. In the theoretical and numerical results that follow, we see that even though the simplest scheme of constant stepsize is considered, the rate of decay of error bound for the ϵ-subgradient algorithm is fundamentally better than the discretization approach as its rate of decay of error bound does not depend on the uncertainty dimension, unlike the discretization case. However, as stated next, we need an additional concavity assumption for the ϵ-subgradient algorithm.

Assumption III.20. *The functions $\phi(\cdot, y)$ are convex for all $y \in Y$, and $\phi(x, \cdot)$ are concave for all $x \in \mathbb{R}^d$.* □

The above assumption is necessary as subgradient algorithms only handle convex problems. In addition, the concavity assumption is required to ensure that the global maximization step in Step 2 of Algorithm III.3 can be completed in finite time. We compare that to the assumptions for discretization algorithms, where strong convexity on $\phi(\cdot, y)$ for all $y \in Y$ is required, but no concavity assumption is necessary. We refer to problems that satisfy Assumption III.20 as convex-concave problems.

We also need the following assumption on the boundedness of the subgradients.

Assumption III.21. *For any bounded set $S \subset \mathbb{R}^d$, there exists an $s < \infty$ such that*

$$\sup_{i \in \mathbb{N}_0, x_i \in S} \{\|g\| | g \in \partial\psi(x_i)\} \leq s. \tag{III.102}$$

□

We obtain the following convergence result for Algorithm III.3 from Bertsekas (2010, p. 349).

Proposition III.22. *Suppose that for all $y \in Y$, $\phi(\cdot, y)$ is convex on $\mathbb{R}^d$. If $\{x_i\}_{i\in\mathbb{N}_0}$ is a sequence generated by Algorithm III.3, then for all $i \in \mathbb{N}_0$ and $x \in \mathbb{R}^d$,*

$$\|x - x_{i+1}\|^2 \leq \|x - x_i\|^2 - 2\alpha\left[\psi(x_i) - \psi(x) - \epsilon\right] + \alpha^2\|g_i\|^2, \tag{III.103}$$

where α and $\epsilon > 0$ are as in Algorithm III.3, and $g_i \in \mathbb{R}^d$ is an ϵ-subgradient of $\psi(\cdot)$ at x_i. □

We denote the optimal solution set of (SMX) by $X^* \stackrel{\triangle}{=} \left\{x \in \mathbb{R}^d | \psi(x) = \psi^*\right\}$ and the distance of the initial point x_0 to X^* by $d(x_0) \stackrel{\triangle}{=} \min_{x\in X^*} \|x_0 - x\|$. We follow the arguments in the convergence analyses of Bertsekas (2010, Section 6.3) on subgradient algorithm to derive the following two convergence results for the ϵ-subgradient algorithm, Algorithm III.3.

Proposition III.23. *Suppose that Assumption III.21 holds, and that $\phi(\cdot, y)$ are convex for all $y \in Y$. If the sequence $\{x_i\}_{i\in\mathbb{N}_0}$ is generated by Algorithm III.3 in solving* (SMX)*, then*

$$\liminf_{i\to\infty} \psi(x_i) \leq \psi^* + \frac{\alpha s^2}{2} + \epsilon, \tag{III.104}$$

where α and $\epsilon > 0$ are as in Algorithm III.3, and s is as in Assumption III.21.

Proof. The proof follows the same arguments as that for the subgradient algorithm in Bertsekas (2010, p. 275), with the difference that (III.103) is used here for the ϵ-subgradient algorithm instead of the corresponding equation for the subgradient algorithm. □

The next result gives an estimate of the number of iterations required by Algorithm III.3 to attain an error tolerance of $(\alpha s^2/2) + \epsilon + \epsilon'/2$, for any $\epsilon' > 0$.

Theorem III.24. *Suppose that the functions $\phi(\cdot, y)$ are convex for all $y \in Y$. Suppose also that Assumption III.21 holds, and the sequence $\{x_i\}_{i\in\mathbb{N}_0}$ is generated by Algorithm III.3 in solving* (SMX)*. If X^* is nonempty, then for any $\epsilon' > 0$,*

$$\min_{0\leq i\leq K} \psi(x_i) - \psi^* \leq \frac{\alpha s^2 + 2\epsilon + \epsilon'}{2}, \tag{III.105}$$

where

$$K = \left\lfloor \frac{d(x_0)^2}{\alpha\epsilon'} \right\rfloor, \tag{III.106}$$

α and $\epsilon > 0$ are as in Algorithm III.3, and s is as in Assumption III.21.

Proof. We follow the proof for the subgradient algorithm in Bertsekas (2010, Proposition 6.3.3), with (III.103) replacing the inequality 6.3.1(a) of Bertsekas (2010). For the sake of contradiction, we assume that (III.105) does not hold. Thus, for all i such that $0 \leq i \leq K$,

$$\psi(x_i) - \psi^* - \epsilon > \frac{\alpha s^2 + \epsilon'}{2}. \tag{III.107}$$

Using this relation in (III.103), with $x \in X^*$, we obtain for all i such that $0 \leq i \leq K$,

$$\begin{aligned} \min_{x^* \in X^*} \|x_{i+1} - x^*\|^2 &\leq \min_{x^* \in X^*} \|x_i - x^*\|^2 - 2\alpha \left[\psi(x_i) - \psi^* - \epsilon\right] + \alpha^2 s^2 \\ &\leq \min_{x^* \in X^*} \|x_i - x^*\|^2 - 2\alpha \frac{\alpha s^2 + \epsilon'}{2} + \alpha^2 s^2 \\ &\leq \min_{x^* \in X^*} \|x_i - x^*\|^2 - \alpha\epsilon'. \end{aligned} \tag{III.108}$$

Applying (III.108) recursively, we obtain

$$0 \leq \min_{x^* \in X^*} \|x_{i+1} - x^*\|^2 \leq \min_{x^* \in X^*} \|x_0 - x^*\|^2 - (K+1)\alpha\epsilon'. \tag{III.109}$$

Solving for K gives

$$K \leq \frac{\min_{x^* \in X^*} \|x_0 - x^*\|^2}{\alpha\epsilon'} - 1 = \frac{d(x_0)^2}{\alpha\epsilon'} - 1, \tag{III.110}$$

which contradicts (III.106). □

The following assumption regarding the computational work required for function and gradient evaluations provide the basis for analyzing the computational work required for Algorithm III.3 to solve (SMX).

Assumption III.25. *There exist constants $\gamma, a', a'' < \infty$ such that for any $x \in \mathbb{R}^d$, $y \in Y \subset \mathbb{R}^m$, the computational work to evaluate any of the three functions $\phi(x, y)$, $\nabla_x \phi(x, y)$, or $\nabla_y \phi(x, y)$ is no larger than $\gamma m^{a'} d^{a''}$.* □

The following assumption ensures that Algorithm III.3 generates bounded sequences.

Assumption III.26. *The level set*

$$L(x_0) \stackrel{\triangle}{=} \{x | \psi(x) \leq \psi(x_0)\} \tag{III.111}$$

is bounded, for all $x_0 \in \mathbb{R}^d$, where x_0 is as in Algorithm III.3. □

We refer to the iterations of Algorithm III.3 as major iterations and the iterations of the algorithm map in Step 2 of Algorithm III.3 as minor iterations. We denote the j^{th} minor iterate during the i^{th} major iteration as $y_{i,j}$.

Assumption III.27. *A linearly convergent algorithm map is used in Step 2 of Algorithm III.3, i.e., there exist a $c \in (0,1)$ such that*

$$\frac{\psi(x_i) - \phi(x_i, y_{i,j+1})}{\psi(x_i) - \phi(x_i, y_{i,j})} \leq c \tag{III.112}$$

for all $j \geq 1$. In addition, the computational work in the linearly converging algorithm is no larger than a constant number of function and gradient evaluations at each iteration. □

We define

$$e_0^{\max} \stackrel{\triangle}{=} \sup_{x \in \mathbb{R}^d, y_1, y_2 \in Y} |\phi(x, y_1) - \phi(x, y_2)|. \tag{III.113}$$

The next result provides an upper bound on the computational work of Algorithm III.3.

Theorem III.28. *Suppose that Assumptions III.2, III.14, III.20, III.25, III.26, III.27 hold, and X^* is nonempty. Then for any $x_0 \in \mathbb{R}^d, \epsilon, \epsilon', \alpha > 0$, there exist constants $a, a', a'', c', c'' < \infty$, such that the computational work in Algorithm III.3 to generate $\{x_i\}_{i=0}^K$, to solve* (SMX)*, while satisfying (III.105) and (III.106) is no larger than*

$$\frac{a}{\alpha \epsilon'} \left[m^{c'} d^{c''} + m^{a'} d^{a''} \left(\frac{\log(\epsilon / e_0^{\max})}{\log c} + 1 \right) \right] \text{ if } \epsilon < e_0^{\max}, \tag{III.114}$$

and

$$\frac{am^{c'}d^{c''}}{\alpha\epsilon'} \text{ otherwise.} \tag{III.115}$$

Proof. The main computational work in Algorithm III.3 is in Step 2, to determine an ϵ-maximizer y_i. Since Y is bounded, and for all $x \in \mathbb{R}^d$, $\phi(x, \cdot)$ is globally Lipschitz continuous according to Assumption III.2, $e_0^{\max} < \infty$. Based on Assumption III.27, the number of iterations required to obtain y_i such that $\psi(x_i) - \phi(x_i, y_i) \leq \epsilon'$ is no larger than $(\log(\epsilon/e_0^{\max})/\log c) + 1$ if $\epsilon < e_0^{\max}$, and equals zero if $\epsilon \geq e_0^{\max}$. Since the computational work in the linearly convergent algorithm is no larger than a constant number of function and gradient evaluations at each iteration, based on Assumption III.25, there exist constants $\gamma, a', a'' < \infty$ such that the computational work at each iteration is no larger than $\gamma m^{a'} d^{a''}$.

The main computational work in Step 3 of Algorithm III.3 is the computation of the gradient $\nabla_x \phi(x_i, y_i)$, and based on Assumption III.25, there exist constants $\zeta, c', c'' < \infty$ such that the computational work for Step 3 is no larger than $\zeta m^{c'} d^{c''}$.

Based on Assumption III.26 and (III.106), there exists a $\beta < \infty$ such that

$$K = \left\lfloor \frac{d(x_0)^2}{\alpha\epsilon'} \right\rfloor \leq \frac{d(x_0)^2}{\alpha\epsilon'} \leq \frac{\beta}{\alpha\epsilon'}. \tag{III.116}$$

Thus the overall computational work for Algorithm III.3 to generate $\{x_i\}_{i=0}^K$, satisfying (III.105) and (III.106) is no larger than

$$\frac{a}{\alpha\epsilon'}\left[m^{c'}d^{c''} + m^{a'}d^{a''}\left(\frac{\log(\epsilon/e_0^{\max})}{\log c} + 1\right)\right] \text{ if } \epsilon < e_0^{\max}, \tag{III.117}$$

where $a = \max\{\beta\zeta, \beta\gamma\}$, and

$$\frac{am^{c'}d^{c''}}{\alpha\epsilon'} \text{ otherwise,} \tag{III.118}$$

where $a = \beta\zeta$. □

From (III.105), we define the error bound for Algorithm III.3 as

$$e_b^{\text{subgrad}} \triangleq \frac{\alpha_b s^2 + 2\epsilon_b + \epsilon_b'}{2}, \tag{III.119}$$

with $\alpha_b, \epsilon_b, \epsilon'_b > 0$ satisfying the following inequalities for $b \in \mathbb{N}$,

$$\frac{a}{\alpha_b \epsilon'_b}\left[m^{c'}d^{c''} + m^{a'}d^{a''}\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\right] \leq b \text{ if } \epsilon_b < e_0^{\max}, \tag{III.120}$$

and

$$\frac{am^{c'}d^{c''}}{\alpha_b \epsilon'_b} \leq b \text{ otherwise.} \tag{III.121}$$

We call a sequence $\{\epsilon_b, \epsilon'_b, \alpha_b\}_{b=1}^{\infty}, b \in \mathbb{N}$ a feasible sequence if it satisfies (III.120) and (III.121).

The next result states that Algorithm III.3 achieves an asymptotic rate of decay of error bound e_b^{subgrad} of $b^{-1/2}$ as $b \to \infty$.

Theorem III.29. *Suppose that Assumptions III.2, III.14, III.20, III.21, III.25, III.26, and III.27 hold. If Algorithm III.3 is used to solve* (SMX)*, then for all arbitrary feasible sequences of* $\{\epsilon_b, \epsilon'_b, \alpha_b\}_{b\in\mathbb{N}}$*,*

$$\liminf_{b\to\infty} \frac{\log e_b^{\text{subgrad}}}{\log b} \geq -\frac{1}{2}. \tag{III.122}$$

For all $b \in \mathbb{N}$*, if* $\epsilon_b = \epsilon'_b = m/\sqrt{b}$ *and if*

$$\alpha_b = \frac{a}{b\epsilon'_b}\left[m^{c'}d^{c''} + m^{a'}d^{a''}\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\right] \text{ whenever } \epsilon_b < e_0^{\max}, \tag{III.123}$$

and

$$\alpha_b = \frac{am^{c'}d^{c''}}{b\epsilon'_b} \text{ otherwise,} \tag{III.124}$$

then

$$\lim_{b\to\infty} \frac{\log e_b^{\text{subgrad}}}{\log b} = -\frac{1}{2}. \tag{III.125}$$

Proof. For any arbitrary feasible sequence of $\{\epsilon_b, \epsilon'_b, \alpha_b\}_{b\in\mathbb{N}}$, we first consider the set $\mathcal{A} \subset \mathbb{N}$, where $\epsilon_b \geq e_0^{\max}$ for all $b \in \mathcal{A}$. Based on (III.119), $e_b^{\text{subgrad}} \geq e_0^{\max}$ for all $b \in \mathcal{A}$. Since $e_0^{\max} \geq 0$ by definition, there exists a $b_0 \in \mathbb{N}$ such that

$$\frac{\log e_b^{\text{subgrad}}}{\log b} \geq -\frac{1}{2} \tag{III.126}$$

for all $b \in \mathcal{A}, b \geq b_0$.

Next, we consider those $b \in \mathbb{N} - \mathcal{A}$ (where '$-$' represents the set difference operator), where $\epsilon_b < e_0^{\max}$. Based on (III.120), we obtain that

$$\alpha_b \geq \frac{a}{b\epsilon'_b}\left[m^{c'}d^{c''} + m^{a'}d^{a''}\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\right]. \tag{III.127}$$

We substitute α_b into (III.119), and we obtain that

$$\begin{aligned} e_b^{\text{subgrad}} &\geq \frac{a}{2b\epsilon'_b}\left[m^{c'}d^{c''} + m^{a'}d^{a''}\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\right]s^2 + \epsilon_b + \frac{\epsilon'_b}{2} \\ &= \frac{am^{c'}d^{c''}s^2}{2b\epsilon'_b} + \frac{am^{a'}d^{a''}s^2}{2b\epsilon'_b}\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right) + \epsilon_b + \frac{\epsilon'_b}{2}. \end{aligned} \tag{III.128}$$

Taking log on both sides, we obtain that

$$\begin{aligned} \log e_b^{\text{subgrad}} &\geq \log\Bigg(\exp\left[\log am^{c'}d^{c''}s^2 - \log 2b\epsilon'_b\right] + \exp\Bigg[\log am^{a'}d^{a''}s^2 - \log 2b\epsilon'_b \\ &\quad + \log\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\Bigg] + \exp\left[\log\epsilon_b\right] + \exp\left[\log\frac{\epsilon'_b}{2}\right]\Bigg) \\ &\geq \log\Bigg(\max\Bigg\{\exp\left[\log am^{c'}d^{c''}s^2 - \log 2b\epsilon'_b\right], \\ &\qquad \exp\left[\log am^{a'}d^{a''}s^2 - \log 2b\epsilon'_b + \log\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\right], \\ &\qquad \exp\left[\log\epsilon_b\right], \exp\left[\log\frac{\epsilon'_b}{2}\right]\Bigg\}\Bigg) \end{aligned}$$

$$\begin{aligned} &= \max\Bigg\{\log\left(\exp\left[\log am^{c'}d^{c''}s^2 - \log 2b\epsilon_b\right]\right), \\ &\qquad \log\left(\exp\left[\log am^{a'}d^{a''}s^2 - \log 2b\epsilon_b + \log\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right)\right]\right), \\ &\qquad \log\left(\exp\left[\log\epsilon_b\right]\right), \log\left(\exp\left[\log\frac{\epsilon'_b}{2}\right]\right)\Bigg\} \\ &= \max\Bigg\{\log am^{c'}d^{c''}s^2 - \log 2b\epsilon_b, \\ &\qquad \log am^{a'}d^{a''}s^2 - \log 2b\epsilon_b + \log\left(\frac{\log(\epsilon_b/e_0^{\max})}{\log c}+1\right), \\ &\qquad \log\epsilon_b, \log\frac{\epsilon'_b}{2}\Bigg\}. \end{aligned} \tag{III.129}$$

Hence, for any $b \in \mathbb{N} - \mathcal{A}, b > 1$, we obtain that

$$\begin{aligned}\frac{\log e_b^{\text{subgrad}}}{\log b} \quad \geq \quad \max \Bigg\{ & \frac{\log am^{c'} d^{c''} s^2}{\log b} - \frac{\log 2b\epsilon_b}{\log b}, \\ & \frac{\log am^{a'} d^{a''} s^2}{\log b} - \frac{\log 2b\epsilon_b}{\log b} + \frac{\log \left(\frac{\log(\epsilon_b / e_0^{\max})}{\log c} + 1 \right)}{\log b}, \\ & \frac{\log \epsilon_b}{\log b}, \frac{\log \frac{\epsilon'_b}{2}}{\log b} \Bigg\}. \end{aligned} \tag{III.130}$$

For the sake of contradiction, we assume that there exists a feasible sequence $\{\epsilon_b, \epsilon'_b, \alpha_b\}_{b \in \mathbb{N} - \mathcal{A}}$ such that

$$\liminf_{b \to \infty} \frac{\log e_b^{\text{subgrad}}}{\log b} < -\frac{1}{2}. \tag{III.131}$$

This implies that there exists a $\delta > 0$, a $b_1 > 1$, and an infinite subsequence $\mathcal{B} \subset \mathbb{N} - \mathcal{A}$ such that

$$\frac{\log e_b^{\text{subgrad}}}{\log b} < -\frac{1}{2} - \delta \tag{III.132}$$

for all $b > b_1, b \in \mathcal{B}$.

From (III.130) and (III.132), we obtain that

$$\frac{\log am^{c'} d^{c''} s^2}{\log b} - \frac{\log 2b\epsilon'_b}{\log b} < -\frac{1}{2} - \delta, \tag{III.133}$$

$$\frac{\log am^{a'} d^{a''} s^2}{\log b} - \frac{\log 2b\epsilon'_b}{\log b} + \frac{\log \left(\frac{\log(\epsilon_b / e_0^{\max})}{\log c} + 1 \right)}{\log b} < -\frac{1}{2} - \delta, \tag{III.134}$$

$$\frac{\log \epsilon_b}{\log b} < -\frac{1}{2} - \delta, \tag{III.135}$$

and

$$\frac{\log \frac{\epsilon'_b}{2}}{\log b} < -\frac{1}{2} - \delta, \tag{III.136}$$

for all $b > b_1, b \in \mathcal{B}$. Based on (III.134) and (III.136), there exists $\delta > 0$, $b_2 > b_1$, and an infinite subsequence $\mathcal{B}$ such that

$$-1 - \frac{\log \epsilon'_b}{\log b} < -\frac{1}{2} - \frac{1}{2}\delta \tag{III.137}$$

and

$$\frac{\log \epsilon_b'}{\log b} < -\frac{1}{2} - \frac{1}{2}\delta \tag{III.138}$$

for all $b > b_2, b \in \mathcal{B}$.

From (III.137), we obtain that

$$\frac{\log \epsilon_b'}{\log b} > -\frac{1}{2} + \frac{1}{2}\delta, \tag{III.139}$$

which contradicts (III.138).

Thus, the conclusion follows for the first part of the theorem.

Next, we prove the second part of the theorem. Since $\epsilon_b = m/\sqrt{b}$, there exists a $b_1 \in \mathbb{N}$ such that $\epsilon_b < e_0^{\max}$. Since we are concerned with the asymptotic rate of decay of error bound e_b^{subgrad} when $b \to \infty$, we only need to consider the case where $\epsilon_b < e_0^{\max}$. Thus, substituting $\epsilon_b = \epsilon_b' = m/\sqrt{b}$ and α_b as in (III.123) into (III.119), we obtain that

$$\begin{aligned} e_b^{\text{subgrad}} &= \frac{a}{2m\sqrt{b}} \left[m^{c'} d^{c''} + m^{a'} d^{a''} \left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c} + 1 \right) \right] s^2 + \frac{m}{\sqrt{b}} + \frac{m}{2\sqrt{b}} \\ &= \frac{1}{\sqrt{b}} \left[\frac{a m^{c'} d^{c''} s^2}{2m} + \frac{a m^{a'} d^{a''} s^2}{2m} \left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c} + 1 \right) + \frac{3m}{2} \right]. \end{aligned} \tag{III.140}$$

Taking logs on both sides of (III.140), we obtain that

$$\log e_b^{\text{subgrad}} = \log \frac{1}{\sqrt{b}} + \log \left[\frac{a m^{c'} d^{c''} s^2}{2m} + \frac{a m^{a'} d^{a''} s^2}{2m} \left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c} + 1 \right) + \frac{3m}{2} \right]. \tag{III.141}$$

We consider the second term in (III.141), and obtain that

$$\lim_{b\to\infty}\log\left[\frac{am^{c'}d^{c''}s^2}{2m}+\frac{am^{a'}d^{a''}s^2}{2m}\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)+\frac{3m}{2}\right]$$
$$=\lim_{b\to\infty}\log\left[\frac{am^{a'}d^{a''}s^2}{2m}\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)\left(\frac{\frac{am^{a'}d^{a''}s^2}{2m}+\frac{3m}{2}}{\frac{am^{a'}d^{a''}s^2}{2m}\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)}+1\right)\right]. \tag{III.142}$$

Based on the continuity of the log function, and the fact that

$$\lim_{b\to\infty}\frac{\frac{am^{a'}d^{a''}s^2}{2m}+\frac{3m}{2}}{\frac{am^{a'}d^{a''}s^2}{2m}\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)}=0, \tag{III.143}$$

the right-hand side of (III.142) simplifies to

$$\lim_{b\to\infty}\log\left[\frac{am^{a'}d^{a''}s^2}{2m}\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)\right]. \tag{III.144}$$

From (III.141) and (III.144), we obtain that

$$\lim_{b\to\infty}\frac{\log e_b^{\text{subgrad}}}{\log b}=\lim_{b\to\infty}\frac{\log\frac{1}{\sqrt{b}}}{\log b}+\lim_{b\to\infty}\frac{\log\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)}{\log b} \tag{III.145}$$

as $\lim_{b\to\infty}\frac{\log\left(\frac{am^{a'}d^{a''}s^2}{2m}\right)}{\log b}=0$. Applying L'Hopital's rule on the second term of the right-hand side, we obtain that

$$\lim_{b\to\infty}\frac{\log\left(\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1\right)}{\log b}=\lim_{b\to\infty}\frac{\left(\frac{1}{\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1}\right)\left(\frac{1}{\frac{m\log c}{e_0^{\max}\sqrt{b}}}\right)\left(-\frac{m}{2e_0^{\max}b^{3/2}}\right)}{1/b}$$
$$=\lim_{b\to\infty}\left(\frac{1}{\frac{\log\left(\frac{m}{e_0^{\max}\sqrt{b}}\right)}{\log c}+1}\right)\left(-\frac{1}{2\log c}\right)=0, \tag{III.146}$$

and the conclusion follows. □

We see from Theorem III.29 that a certain choice of the parameters ϵ_b, ϵ'_b, and α_b, $b \in \mathbb{N}$, results in an asymptotic rate of decay of error bound e_b^{subgrad} of $b^{-1/2}$. If we compare against the fastest rate of decay for a discretization algorithm of $b^{-1/m\nu}$, we see that for moderate and large m, discretization algorithms may not be competitive against Algorithm III.3. This difference in the rate of decay of error bound is observed in the numerical results in the next section.

D. NUMERICAL RESULTS

In this section, we provide numerical evidence to validate some of the key theoretical results obtained in Sections III.B and III.C. Proposition III.7 indicates that the asymptotic rate of decay of error bounds for discretization algorithms is no faster than $b^{-1/m\nu}$ as $b \to \infty$. We compare that with the ϵ-subgradient algorithm, Algorithm III.3, where Theorem III.29 indicates that a rate of $b^{-1/2}$ as $b \to \infty$ is attainable. The dependence on m, the uncertainty dimension for the discretization algorithms, implies that under certain convexity-concavity assumptions, discretization algorithms will likely not be competitive against ϵ-subgradient algorithms for semi-infinite problems with high uncertainty dimension. In this section, we provide some indication on the range of values of m where discretization algorithms are not competitive with ϵ-subgradient algorithms for convex-concave problems.

From Theorems III.9 and III.11, the asymptotic rate of decay of error bound for a discretization algorithm that uses a quadratically convergent algorithm map is the same as that of a linearly convergent algorithm map. In this section, we examine if there is any numerical difference between a superlinearly convergent algorithm map and a linearly convergent algorithm map.

We compare the following algorithms over a set of problem instances from Rustem and Howe (2002):

(i) Algorithm III.1 with ϵ-PPP (an active-set version of PPP as stated in Algorithm 2.4.34 in Polak 1997; see also Polak 2008) as the algorithm map.

(ii) Algorithm III.1 with SQP-2QP (Algorithm 2.1 of Zhou & Tits 1996, an SQP algorithm with two QPs) as the algorithm map.

(iii) The ϵ-subgradient algorithm, Algorithm III.3.

The first two algorithms are discretization algorithms, while the third is an ϵ-subgradient algorithm. We refer to Appendix D for details on the algorithms and the algorithm parameters used.

We use Problems 1 and 5 from Rustem and Howe (2002, pp. 100-102), which are two- and three-dimensional in y, respectively. We also modify Problem 1 to create a one-dimensional (in y) problem instance. We call the three problem instances SProbA, SProbB, and SProbC, in increasing order of y-dimensionality; see Appendix C for details.

Similar to Chapter II, we implement and run all algorithms in MATLAB version 7.7.0 (R2008b) (see Mathworks, 2009) on a 3.73 GHz PC using Windows XP SP3, with 3 GB of RAM. All QPs are solved using TOMLAB CPLEX version 7.0 (R7.0.0) (see Tomlab, 2009) with the Primal Simplex option. In Step 2 of Algorithm III.3, we use TOMLAB SNOPT version 7.2-5 (see Gill, Murray, & Saunders, 2007) to find the y-maximizer.

In Chapter II, we consider problem instances with uncertainty dimension of one, and we use discretization parameters N in the order of 10^6 to achieve reasonable solution tolerance. In all the finite minimax algorithms considered (including SQP-2QP and ϵ-PPP), one of the steps is to compute the function values at all the grid points at the current iterate. In Chapter II, we implement the function evaluation step using vector operations on all the grid points in a single line of code, instead of "looping" through each grid point, to ensure better efficiency. In this chapter, we consider problems with uncertainty dimensions higher than one, and the discretization parameters required to achieve reasonable solution tolerance increase to orders of 10^8 and above. This requires too much memory if the same implementation as that in

Chapter II is used. Thus, we use "looping," evaluating subsets of the grid points in each loop. The different implementation is applied for SProbB ($m = 2$) and SProbC ($m = 3$), since the original more efficient implementation still works for SProbA ($m = 1$). Note that this issue does not affect ϵ-subgradient algorithms as they do not deal with grid points.

We report run times to achieve a solution x that satisfies

$$\|x - x^{\text{target}}\| \leq t, \tag{III.147}$$

where the error tolerance $t = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$, and x^{target} is a target solution (see Table 18 in Appendix C) obtained by Algorithm III.3. We refer to Appendix C for details on the procedure to obtain x^{target}. Algorithm III.3 is chosen for these verification analysis as preliminary experiments show that it is significantly more efficient than the other two algorithms, especially for problems with uncertainty dimension $m \geq 2$. Although the termination criterion (III.147) is not possible for real-world problems, as x^{target} is usually unknown beforehand, we find that it is the most useful criterion in this study.

1. Problem Instance of Uncertainty Dimension One

Table 8 summarizes the run times (in seconds) of Algorithm III.1 with ϵ-PPP, for various discretization parameter N_b across the top row, to achieve various error tolerances t listed in the first column. Run times in boldface indicate the particular discretization parameter N_b that produces the shortest run time for the specific error tolerance t. An asterisk * in the table indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance. For example, in Table 8 with $N_b = 1,000$, we observe that the iterates do not change after a certain time (within six hours), and the required error tolerance of $t = 10^{-4}$ has not been met. A double asterisks ** indicate that the algorithm failed to satisfy the required error tolerance after six hours. Preliminary experiments show that Algorithm III.3 produces run times no slower than ten seconds for all problem instances considered. Thus,

we choose an arbitrary maximum run time of six hours (significantly longer than ten seconds). As mentioned, the MATLAB implementation for ϵ-PPP on SProbA computes the function values in a single line of code. As there is insufficient memory to store the function values of all N_b =100,000,000 functions, that leads to the memory issues as indicated by "mem" in Table 8.

$t\backslash N_b$	100	1,000	10,000	100,000	1,000,000	10,000,000	100,000,000
10^{-1}	0.83 (7)	**0.57 (7)**	0.66 (7)	1.6 (7)	11.4 (7)	107.6 (7)	mem
10^{-2}	0.84 (10)	**0.77 (10)**	0.98 (10)	2.2 (10)	15.8 (10)	149.0 (10)	mem
10^{-3}	*	2.2 (13)	**1.5 (12)**	3.9 (12)	22.4 (12)	207.1 (12)	mem
10^{-4}	*	*	**4.5 (15)**	9.1 (15)	36.7 (14)	334.7 (14)	mem
10^{-5}	*	*	*	**	**	**	mem

Table 8. Run times (in seconds) for SProbA using Algorithm III.1 with ϵ-PPP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours. The word "mem" means that the algorithm terminates due to insufficient memory.

Table 9 summarizes the run times of Algorithm III.1 with SQP-2QP. The faster run times in Table 9 compared to Table 8 are due to the superlinear rate of convergence of the SQP-2QP algorithm map compared to the linear rate of convergence of ϵ-PPP.

We see from Tables 8 and 9, as well as subsequent run times for SProbB and SProbC that the discretization parameter N_b that produces the fastest run times, varies between problems and tolerances. Thus, it is difficult to determine the "right" discretization parameter to use.

Table 10 summarizes the run times of Algorithm III.3 for SProbA. Comparing the run times for the three algorithms (ignoring the issue that discretization parameters are difficult to determine), we see that the discretization algorithms (Tables 8 and 9) are generally competitive against the ϵ-subgradient algorithm (Table 10) for problems with uncertainty dimension of one.

$t\backslash N_b$	100	1,000	10,000	100,000	1,000,000	10,000,000	100,000,000
10^{-1}	0.12 (5)	0.14 (5)	**0.11 (5)**	0.51 (5)	4.4 (5)	42.6 (5)	mem
10^{-2}	0.17 (6)	0.18 (6)	**0.12 (6)**	0.59 (6)	5.1 (6)	48.8 (6)	mem
10^{-3}	*	0.15 (7)	**0.13 (7)**	0.65 (7)	5.7 (7)	55.1 (7)	mem
10^{-4}	*	*	**0.13 (8)**	0.81 (8)	6.4 (8)	61.4 (8)	mem
10^{-5}	*	*	*	*	*	*	mem

Table 9. Run times (in seconds) for SProbA using Algorithm III.1 with SQP-2QP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance. The word "mem" means that the algorithm terminates due to insufficient memory.

t	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Run times	0.18 (2)	0.25 (3)	0.36 (4)	0.34 (5)	0.45 (6)

Table 10. Run times (in seconds) for SProbA using Algorithm III.3. The numbers in parentheses indicate the number of iterations.

2. Problem Instance of Uncertainty Dimension Two

Tables 11-13 summarize the run times for SProbB. The discretization parameter N_b is chosen such that $N_b^{1/m} \in \mathbb{N}$. The run times for the two discretization algorithms are generally an order of magnitude slower than those for SProbA, while the run times for Algorithm III.3 are still within the same order of magnitude. These results provide some validation to the $b^{-1/m\nu}$ rate of decay of error bound obtained for the two discretization algorithms in Theorems III.9 and III.11, and the $b^{-1/2}$ rate for Algorithm III.3 in Theorem III.29.

3. Problem Instance of Uncertainty Dimension Three

Tables 14-16 summarize the run times for SProbC. We see more evidence of the independence of the rate of decay of error bound on m for Algorithm III.3. Specifically, the ratio of run times for Algorithm III.3 to attain error tolerances of 10^{-1} and 10^{-5} are $0.45/0.18 = 2.5$ (SProbA where $m = 1$), $1.1/0.21 = 5.2$ (SProbB where $m = 2$), and $4.0/1.6 = 2.5$ (SProbC where $m = 3$). These ratios provide some

$t\backslash N_b$	1,024	10,000	100,489	1,000,000	10,004,569	100,000,000	1,000,014,129
10^{-1}	2.5 (5)	**1.5 (5)**	4.9 (5)	29.6 (5)	281.9 (5)	2720 (5)	**
10^{-2}	*	**2.0 (6)**	6.2 (7)	39.6 (7)	380.8 (7)	3605 (7)	**
10^{-3}	*	*	*	**49.9 (9)**	535.1 (9)	4720 (9)	**
10^{-4}	*	*	*	**49.4 (9)**	670.1 (10)	6899 (11)	**
10^{-5}	*	*	*	*	*	**	**

Table 11. Run times (in seconds) for SProbB using Algorithm III.1 with ϵ-PPP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours.

$t\backslash N_b$	1,024	10,000	100,489	1,000,000	10,004,569	100,000,000	1,000,014,129
10^{-1}	0.72 (4)	**0.46 (4)**	2.3 (5)	12.4 (4)	134.4 (5)	1146 (4)	10879 (5)
10^{-2}	*	**0.53 (5)**	2.5 (6)	14.4 (5)	155.2 (6)	1346 (5)	12475 (6)
10^{-3}	*	*	*	**16.7 (6)**	175.6 (7)	1532 (6)	13881 (7)
10^{-4}	*	*	*	*	*	**1719 (7)**	15178 (8)
10^{-5}	*	*	*	*	*	*	*

Table 12. Run times (in seconds) for SProbB using Algorithm III.1 with SQP-2QP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance.

t	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Run times	0.21 (5)	0.35 (7)	0.40 (8)	0.71 (9)	1.1 (9)

Table 13. Run times (in seconds) for SProbB using Algorithm III.3. The numbers in parentheses indicate the number of iterations.

validation to Theorem III.29, which states that the asymptotic rate of decay of error bound for Algorithm III.3 is $b^{-1/2}$, which is independent of m.

For discretization algorithms, we see that the increase is strongly dependent on m. For Algorithm III.1 with ϵ-PPP, the ratio of run times to attain error tolerances of 10^{-1} and 10^{-4} are $4.5/0.57 = 7.9$ (SProbA), $49.4/1.5 = 32.9$ (SProbB), and $> 21,600/1.4 = 15,000$ (SProbC). For Algorithm III.1 with SQP-2QP, the ratio of run times to attain error tolerances of 10^{-1} and 10^{-4} are $0.13/0.11 = 1.2$ (SProbA), $1,719/0.46 = 3,737$ (SProbB), and $> 21,600/0.81 = 26,667$ (SProbC). These observations indicate that the additional computational work to achieve smaller errors increases as m increases, which again provides validation to Theorems III.9 and III.11, which states that the asymptotic rate of decay of error bounds for Algorithm III.1 with ϵ-PPP and SQP-2QP are $b^{-1/m\nu}$.

Theorems III.9 and III.11 state that the error bounds for the discretization algorithms with a quadratically and linearly convergent algorithm map decay at the same asymptotic rate of $b^{-1/m\nu}$ as $b \to \infty$. We observe generally faster run times for Algorithm III.1 with SQP-2QP (superlinear) as compared to ϵ-PPP (linear), which shows that we are not in asymptotic regime yet.

$t\backslash N_b$	1,000	10,648	103,823	1,000,000	10,077,696	100,544,625	1,000,000,000
10^{-1}	**1.4 (5)**	3.0 (4)	13.3 (4)	74.4 (4)	491.3 (4)	3706 (4)	**
10^{-2}	*	*	**20.8 (7)**	113.0 (7)	860.5 (8)	6017 (7)	**
10^{-3}	*	*	*	**	**2393 (13)**	10546 (10)	**
10^{-4}	*	*	*	**	**	**	**
10^{-5}	*	*	*	**	**	**	**

Table 14. Run times (in seconds) for SProbC using Algorithm III.1 with ϵ-PPP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours.

$t\backslash N_b$	1,000	10,648	103,823	1,000,000	10,077,696	100,544,625	1,000,000,000
10^{-1}	**0.81 (5)**	2.2 (5)	11.3 (5)	59.6 (5)	429.9 (5)	4871 (5)	**
10^{-2}	*	*	**12.9 (6)**	69.5 (6)	570.7 (6)	5658 (6)	**
10^{-3}	*	*	*	*	**665.7 (8)**	7052 (8)	**
10^{-4}	*	*	*	*	*	*	**
10^{-5}	*	*	*	*	*	*	**

Table 15. Run times (in seconds) for SProbC using Algorithm III.1 with SQP-2QP. The numbers in parentheses indicate the number of iterations. An asterisk * indicates that the particular discretization parameter is insufficient to achieve the desired error tolerance, while a double asterisk ** indicates that (III.147) is not satisfied after six hours.

t	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Run times	1.6 (13)	1.8 (21)	2.7 (29)	3.9 (37)	4.0 (44)

Table 16. Run times (in seconds) for SProbC using Algorithm III.3. The numbers in parentheses indicate the number of iterations.

E. CONCLUSIONS FOR SEMI-INFINITE MINIMAX

This chapter focuses on the discretization approach to solve unconstrained semi-infinite minimax problems. We develop and compare rate-of-convergence results for various fixed and adaptive discretization algorithms, as well as an ϵ-subgradient algorithm. We present a novel way of expressing rate of convergence, in terms of computational work instead of the typical number of iterations. We show that a fixed discretization algorithm can achieve the same asymptotic convergence rate attained by an adaptive discretization algorithm. We also show that under certain convexity-concavity assumptions, the rates of convergence for discretization algorithms depend on the uncertainty dimension, while the rate of convergence for an ϵ-subgradient algorithm is independent of the uncertainty dimension. This indicates that under convexity-concavity assumptions, discretization algorithms are not likely to be competitive with ϵ-subgradient algorithms for problems with large uncertainty dimension.

Numerical results show that for convex-concave problems, discretization algorithms are not competitive with ϵ-subgradient algorithms for problems with uncertainty dimension as small as two.

IV. SEMI-INFINITE MIN-MAX-MIN PROBLEM

A. INTRODUCTION

In this chapter, we consider a generalized semi-finite min-max-min problem of the form

$$\text{(GMXM)} \quad \min_{x \in X} \psi(x), \tag{IV.1}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\psi(x) \stackrel{\triangle}{=} \max_{y \in Y} \min_{z \in Z(x,y)} \phi(x, y, z), \tag{IV.2}$$

$X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^m$ are compact sets, the set-valued function $Z : \mathbb{R}^d \times \mathbb{R}^m \to 2^{\mathbb{R}^s}$ is continuous (see Section 5.3 of Polak, 1997 for a definition on the continuity of a set-valued function) as well as compact- and nonempty-valued on $X \times Y$, and for all $(x, y) \in X \times Y$ and $z \in Z(x, y)$, $\phi(\cdot, \cdot, \cdot)$ is continuous at (x, y, z). In particular, we focus on the special case where $Z(\cdot, \cdot)$ is a constant set $Z \subset \mathbb{R}^s$, but also deal with the generalized $Z(\cdot, \cdot)$ case. Throughout the chapter, we refer to the case with constant set Z as the *constant* Z case, and the case of the set-valued function $Z(\cdot, \cdot)$ as the *variable* Z case. We denote the semi-infinite min-max-min problem for the constant Z case by (SMXM). Also, we refer to (SMXM) and (GMXM) collectively as (MXM) for brevity.

Applications involving min-max-min optimization include floorplan sizing in electronic circuit boards (Chen & Fan, 1998), obstacle avoidance for robots (Kirjner-Neto & Polak, 1998), optimal design centering, tolerancing and tuning problem (Tits, 1985), geometric facility location problem (Cardinal & Langerman, 2006), and network interdiction problem (Martin, 2007), of which we will give an example.

The problem (MXM) is difficult to solve due to the layers of min and max operators, and, as shown in Ralph and Polak (2000), $\psi(\cdot)$ may not have directional derivatives even when $\phi(\cdot, \cdot, \cdot)$ is smooth. This implies that defining suitable optimal-

ity conditions is difficult. These difficulties have resulted in a rather limited literature on (SMXM), and so far, there is no solution approach for (GMXM).

Ralph and Polak (2000) propose an approach that deals with (SMXM), mainly with $X = \mathbb{R}^d$. The assumptions are (i) X, Y, and Z are compact, and (ii) for all $(y, z) \in Y \times Z$, $\phi(\cdot, y, z)$ is continuously differentiable on X. The authors first discretize Y and Z into Y_N and Z_M, where $N, M \in \mathbb{N}$ are the cardinality of Y_N and Z_M, respectively. Then a master algorithm is used to solve sequences of discretized min-max-min problems of increasing level of discretization. The finite min-max-min algorithm map used to solve the discretized problems (within the master algorithm) applies a method that combines an Armijo-type line search and a trust region approach. The authors then discuss how an exact penalization method can be used to eliminate constraints defining X, if any are present. The main challenge in the approach is, in each iteration of the algorithm map, we need to solve M^N linear programs to determine the search direction. As noted in the paper, this is expected to be a highly computationally intensive task. There are no numerical results in this paper.

In Ralph and Polak (2000), we find another approach for (SMXM) with $X = \mathbb{R}^d$, where the same assumptions and initial discretization step in Ralph and Polak (2000) are used. The author then applies exponential smoothing (as described in Chapter II) to the innermost minimization problem to obtain a finite minimax problem. The algorithm proposed also consists of a master algorithm that solves sequences of the finite minimax problems with increasing level of discretization, using the Pshenichnyi-Pironneau-Polak (PPP) minimax algorithm map. The main challenge in this algorithm is, as the level of discretization increases, there are more functions in Z_M. In order to keep the smoothing error small, the smoothing parameter needs to increase, which may lead to ill-conditioning. Again, there are no numerical results in this paper to provide any hint on the numerical performance of the algorithm.

This chapter proposes a novel approach to solve (MXM). We assume that (i) X and Z are compact and convex, and (ii) for all $y \in Y$, $\phi(\cdot, y, \cdot)$ is continuously differentiable and convex on $X \times Z$. We discretize Y into Y_N to obtain a discretized min-max-min problem, which we then reformulate into a discretized min-min-max problem of larger dimensionality. Finally, we observe that the discretized min-min-max problem can be interpreted as a constrained finite minimax problem. Under our convexity assumptions, we show that for any $N \in \mathbb{N}$, if we solve the constrained finite minimax problem, we obtain a global minimizer of the discretized min-max-min problem. And if the level of discretization N increases to infinity, the points constructed approach the global minimizer of (MXM). The algorithms in Ralph and Polak (2000) and Polak (2003) do not guarantee convergence to a global minimizer even under our convexity assumptions. The main challenge in our approach is the size of the constrained finite minimax problem constructed, which has N functions and $d + Ns$ variables, where d and s are the dimensionality of X and Z, respectively.

We find similar conversion from a min-max-min problem to a min-min-max problem in Martin (2007) for the case with binary variables in the outer min-max, where the min-min-max problem provides a lower bound on the optimal objective-function value. Another possible way of converting a min-max-min problem into a min-min-max problem is to use von Neuman's minimax theorem (see for example, Theorem 5.5.5 of Polak, 1997), but this requires that the sets Y and Z be compact, convex, and constant, and for all $x \in X$ and $z \in Z$, $\phi(x, \cdot, z)$ is concave on Y, and for all $x \in X$ and $y \in Y$, $\phi(x, y, \cdot)$ is convex on Z.

The next section shows that (MXM) arises in network interdiction problems. Section C outlines a new approach to solve (MXM). We obtain some numerical results in Section D by applying the approach on a network interdiction problem. Section E concludes the chapter.

B. DEFENDER-ATTACKER-DEFENDER EXAMPLE

In this section, we describe a defender-attacker-defender (DAD) network interdiction problem. We provide two different formulations for the problem, one in the form of (SMXM) and another in the form of (GMXM).

We consider a network G with node set V and arc set E. A node $u \in V$ can provide nonnegative supply up to a maximum of $ubsupply_u$ at the cost of $costsupply_u$ per unit of supply. Node u requires also a given $demand_u$ of supplies. A defender first decides how much supply to place at node u, which we denote by $SUPPLY_u$.

Second, an attacker decides on the quantity of sorties, $SORTIE_{u,v}$ to attack each arc $(u, v) \in E$, subject to a maximum of *totalsorties*, with the intent to maximize the defender's cost to be defined later. We use an exponential damage function; see for example Nugent (1969); Capps (1970), to model the capacity reduction of an arc that is attacked. We consider $SORTIE_{u,v}$ as a continuous variable as we assume that aircraft carry bomb loads that can be distributed in any way over several arcs. Note that our proposed approach can handle integer restrictions on the decision variables for the maximization in (MXM), which is $SORTIE_{u,v}$ in the DAD problem.

Third, the defender sends flow of supplies between nodes in an attempt to meet demand. The parameters $lbflow_{u,v}$ and $ubflow_{u,v}$ represent the lower and upper bounds on the flow across arc (u, v) before the attack, and $vulcap_{u,v}$ represents the amount of capacity vulnerable to attack for arc (u, v). Based on Nugent (1969); Capps (1970), the remaining capacity of arc (u, v) after the attack is:

$$ubflow_{u,v} - vulcap_{u,v}\left[1 - \exp(-vul_{u,v}SORTIE_{u,v})\right], \tag{IV.3}$$

where $vul_{u,v}$ represents the vulnerability of arc (u, v). A larger value of $vul_{u,v}$ represents that the arc is more vulnerable to attacks. The vulnerability parameter $vul_{u,v}$ indicates the efficiency of a sortie against arc (u, v).

The objective function in the problem is the sum of (i) the cost to place supply at nodes and (ii) the cost to send flow through the network after the attack to satisfy demands. We model the nonlinear effects of congestion on the cost of sending flow

(see for example p. 651 of Ahuja, Magnanti, & Orlin, 1993) across arc (u, v) by

$$\frac{costflow_{u,v} FLOW_{u,v}}{ubflow_{u,v} - FLOW_{u,v} + \epsilon}, \tag{IV.4}$$

where $\epsilon > 0$ is a small number to ensure that the cost of flow remains bounded as $FLOW_{u,v}$ approaches $ubflow_{u,v}$.

In this problem, we assume perfect information, i.e., both the defender and attacker know the full characteristics of the network in terms of bounds on the flow on each arc, the vulnerability of each arc, etc. We also assume that the defender knows the maximum sorties that the attacker can launch, the attacker knows where the supplies are placed before launching the sorties, and finally, the defender knows the remaining capacity of all the arcs in the network before sending flow to satisfy the demands.

We provide the formulation in both forms of (SMXM) and (GMXM) next, with detailed explanation following the model descriptions. In (SMXM), the feasible region of the inner minimization problem must be independent of the decision variables for the outer minimization and the maximization parts. Hence, the capacity and balance of flow constraints are accounted for, by using penalty terms in the objective function. In the case of (GMXM), capacity and balance of flow are imposed as constraints.

Indices

$u \in V$	node (alias v)
$(u, v) \in E$	arc directed from node u to node v

Data

$costsupply_u$	cost to place supply at node u
$costflow_{u,v}$	cost coefficient for flow between nodes u and v
$demand_u$	demand at node u
ϵ	a small number that ensures bounded flow cost
$lbflow_{u,v}$	nonnegative lower bound on flow on arc (u, v)
$penbal$	penalty parameter for violation of flow balance constraints
$pencap$	penalty parameter for violation of capacity constraints
$totalsorties$	total number of attacker sorties the attacker can fly

$ubflow_{u,v}$	upper bound on flow on arc (u, v)
$ubsupply_u$	upper bound on supplies at node u
$vul_{u,v}$	vulnerability of arc (u, v)
$vulcap_{u,v}$	amount of capacity of arc (u, v) vulnerable to attack

Decision Variables

$EXCESSSUPPLY_u$	recourse supply removed from node u
$EXTRASUPPLY_u$	recourse supply placed at node u
$FLOW_{u,v}$	flow from node u to node v
$SORTIE_{u,v}$	number of sorties to attack arc (u, v)
$SUPPLY_u$	amount of supply to be placed at node u

We denote the vector that contains all the components $FLOW_{u,v}, (u, v) \in E$ by *FLOW*, and similarly for *EXCESSSUPPLY*, *EXTRASUPPLY*, *SORTIE*, and *SUPPLY*.

DAD Formulation of the form (SMXM): constant Z case

$$\min_{SUPPLY} \left\{ \begin{array}{l} \max_{SORTIE} \left\{ \begin{array}{l} \min_{FLOW} \sum_{u \in V} costsupply_u SUPPLY_u + \sum_{(u,v) \in E} \dfrac{costflow_{u,v} FLOW_{u,v}}{ubflow_{u,v} - FLOW_{u,v} + \epsilon} \\ + \sum_{u \in V} penbal_u \left[\begin{array}{l} \sum_{v:(u,v) \in E} FLOW_{u,v} - \sum_{v:(v,u) \in E} FLOW_{v,u} \\ - SUPPLY_u + demand_u \end{array} \right]^2 \\ + \sum_{(u,v) \in E} pencap_{u,v} \left[\max \left\{ \begin{array}{l} 0, \\ FLOW_{u,v} - ubflow_{u,v} + \\ vulcap_{u,v} \left[1 - \exp(-vul_{u,v} SORTIE_{u,v}) \right] \end{array} \right\} \right]^2 \\ \text{s.t. } lbflow_{u,v} \leq FLOW_{u,v} \leq ubflow_{u,v} \quad \forall (u,v) \in E \end{array} \right\} \\ \text{s.t. } \sum_{(u,v) \in E} SORTIE_{u,v} \leq totalsorties \\ \quad SORTIE_{u,v} \geq 0 \quad \forall (u,v) \in E \end{array} \right\}$$

$$\text{s.t. } 0 \leq SUPPLY_u \leq ubsupply_u \quad \forall u \in V$$

DAD Formulation of the form (GMXM): variable Z case

$$
\min_{SUPPLY} \left\{ \begin{array}{l} \max_{SORTIE} \left\{ \begin{array}{l} \min_{\substack{FLOW, EXCESSSUPPLY, \\ EXTRASUPPLY}} \sum_{u \in V} costsupply_u SUPPLY_u + \sum_{(u,v) \in E} \frac{costflow_{u,v} FLOW_{u,v}}{ubflow_{u,v} - FLOW_{u,v} + \epsilon} \\ \qquad + \sum_{u \in V} penbal_u EXCESSSUPPLY_u + \sum_{u \in V} penbal_u EXTRASUPPLY_u \\ \text{s.t.} \sum_{v:(u,v) \in E} FLOW_{u,v} - \sum_{v:(v,u) \in E} FLOW_{v,u} = SUPPLY_u - demand_u \\ \qquad - EXCESSSUPPLY_u + EXTRASUPPLY_u \quad \forall u \in V \\ lbflow_{u,v} \leq FLOW_{u,v} \\ \qquad \leq ubflow_{u,v} - vulcap_{u,v} [1 - \exp(-vul_{u,v} SORTIE_{u,v})] \quad \forall (u,v) \in E \\ EXCESSSUPPLY_u \geq 0 \quad \forall u \in V \\ EXTRASUPPLY_u \geq 0 \quad \forall u \in V \end{array} \right\} \\ \text{s.t.} \sum_{(u,v) \in E} SORTIE_{u,v} \leq totalsorties \\ \quad SORTIE_{u,v} \geq 0 \quad \forall (u,v) \in E \end{array} \right\} \\ \text{s.t.} \quad 0 \leq SUPPLY_u \leq ubsupply_u \quad \forall u \in V
$$

Discussion

The main differences between the two formulations lie in the way that the capacity and balance of flow constraints are modeled. In the variable Z case, we model them explicitly. The dummy variables $EXCESSSUPPLY$ and $EXTRASUPPLY$ are included to ensure that the model remains feasible even if the sorties have reduced the capacity of the network to a level where (i) excess supply cannot flow out from a node or (ii) demand at a node is not fully satisfied. And if that happens, we penalize the violation based on (i) the excess supply that needs to be removed or (ii) the extra supply required to satisfy demand fully.

For the constant Z formulation, we model the capacity and balance of flow constraints by including penalty cost terms in the objective function. This allows us to have the inner constraint set being a constant set defined by $0 \leq lbflow_{u,v} \leq FLOW_{u,v} \leq ubflow_{u,v}$ for all arcs $(u,v) \in E$.

The objective functions for both cases express the sum of (i) the cost of placing supply at supply nodes, (ii) cost of sending flow through the network after the attack to satisfy demands, considering congestion effects, and (iii) penalty terms for violation of capacity or balance of flow constraints. The other constraints are self-explanatory.

We note that if we view $SORTIE_{u,v}$ as constants instead of decision variables, the objective functions and feasible sets in both formulations are convex, and the

feasible region is defined by linear functions and box constraints. We will revisit this issue when we discuss our new approach to solve (MXM).

C. APPROACH TO SOLVE THE MIN-MAX-MIN PROBLEM

In this section, we propose an approach to solve (MXM) by constructing a constrained finite minimax problem. We use this finite minimax problem to obtain an approximation to a global minimizer of (MXM) under certain assumptions on the algorithm used to solve the finite minimax problem.

Before we describe the approach to solve (MXM), we state two assumptions for Z that will be used repeatedly throughout the chapter. Assumption IV.1 will be used when we consider the constant Z case, while Assumption IV.2 will be used when we consider the variable Z case.

Assumption IV.1. *$X \subset \mathbb{R}^d$, $Y \subset \mathbb{R}^m$, and $Z \subset \mathbb{R}^s$ are compact sets, and $\phi(\cdot,\cdot,\cdot)$ is continuous on $X \times Y \times Z$.*

Assumption IV.2. *$X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^m$ are compact sets, the set-valued function $Z : \mathbb{R}^d \times \mathbb{R}^m \to 2^{\mathbb{R}^s}$ is continuous as well as compact- and nonempty-valued on $X \times Y$, and for all $(x, y) \in X \times Y$ and $z \in Z(x, y)$, $\phi(\cdot,\cdot,\cdot)$ is continuous at (x, y, z).*

1. Constructing a Finite Minimax Problem

In this subsection, we construct a finite minimax problem from (MXM). We first discretize the set $Y \subset \mathbb{R}^m$ to obtain a discretized min-max-min problem. Next, we show that the inner max-min problem is equivalent to a min-max problem. We then observe that the min-min-max problem can be interpreted as a finite minimax problem, but with more variables than (MXM). We cover the details next.

a. Discretized Min-Max-Min Problem

We introduce the discretized problem for (GMXM):

$$(\text{GMXM}_N) \quad \min_{x \in X} \psi_N(x), \tag{IV.5}$$

where $\psi_N : \mathbb{R}^d \to \mathbb{R}, N \in \mathbb{N}$ is defined by

$$\psi_N(x) \stackrel{\triangle}{=} \max_{y \in Y_N} \min_{z \in Z(x,y)} \phi(x, y, z), \tag{IV.6}$$

$Y_N \subset Y$, $|Y_N| = N \in \mathbb{N}$, satisfy the property $\text{dist}(Y_N, Y) \to 0$ as $N \to \infty$, with $\text{dist}(\cdot, \cdot)$ being the Hausdorff distance operator defined on p. 65. Under Assumption IV.2, (GMXM_N) is well-defined. An example of a discretization scheme that produces Y_N with the above property is the uniform grid discussed on p. 65 for the case when Y is a hyper-box. We denote the equivalent discretized problem for (SMXM) by (SMXM_N).

We need the following notation for subsequent analysis of (GMXM):

$$\omega(x, y) \stackrel{\triangle}{=} \min_{z \in Z(x,y)} \phi(x, y, z), \tag{IV.7}$$

$$\hat{Z}(x, y) \stackrel{\triangle}{=} \{z \in Z(x, y) \mid \phi(x, y, z) = \omega(x, y)\}, \tag{IV.8}$$

$$\hat{Y}(x) \stackrel{\triangle}{=} \{y \in Y \mid \omega(x, y) = \psi(x)\}, \tag{IV.9}$$

and

$$\hat{Y}_N(x) \stackrel{\triangle}{=} \{y \in Y_N \mid \omega(x, y) = \psi_N(x)\}. \tag{IV.10}$$

We next show the continuity of the functions $\omega(\cdot, \cdot)$, $\psi(\cdot)$, and $\psi_N(\cdot)$ for both the constant and variable Z case.

Proposition IV.3. *Suppose that Assumption IV.1 holds. Then the functions $\omega(\cdot, \cdot)$, $\psi(\cdot)$, and $\psi_N(\cdot), N \in \mathbb{N}$ are continuous on $X \times Y$ and X, respectively.*

Proof. We refer to Polak (1997, Section 5.4) for the proof. □

Proposition IV.4. *Suppose that Assumption IV.2 holds. Then the functions $\omega(\cdot, \cdot)$, $\psi(\cdot)$, and $\psi_N(\cdot), N \in \mathbb{N}$ are continuous on $X \times Y$ and X, respectively.*

Proof. Based on Corollary 5.4.2 of Polak (1997), $\omega(\cdot, \cdot)$ is continuous on $X \times Y$. Hence, based on Propositions I.1 and I.2, $\psi(\cdot)$ and $\psi_N(\cdot)$ are also continuous on X. □

We next show that the discretized min-max-min problems epi-converge to (MXM). We first consider the constant Z case.

Lemma IV.5. *Suppose that Assumption IV.1 holds. The sequence of problems* $\{(\text{SMXM}_N)\}_{N\in\mathbb{N}}$ *epi-converges to* (SMXM) *as* $N \to \infty$.

Proof. The proof follows the same arguments as the epi-convergence proof in Theorem 5.2 of Polak (2003) and is included here for completeness. Suppose that $\{x_N\}_{N\in\mathbb{N}}$ is a sequence in X such that $x_N \to \hat{x}$ as $N \to \infty$, and suppose that $y_N \in \hat{Y}_N(x_N)$ for each $N \in \mathbb{N}$. Without loss of generality, we assume that $y_N \to \hat{y}$ as $N \to \infty$. Then

$$\limsup_{N\to\infty} \psi_N(x_N) = \lim_{N\to\infty} \omega(x_N, y_N) = \omega(\hat{x}, \hat{y}) \leq \psi(\hat{x}), \tag{IV.11}$$

where we use the fact that $\omega(\cdot,\cdot)$ is continuous; see Proposition IV.3.

Next, suppose that $\psi(\hat{x}) = \omega(\hat{x}, y^*)$ for some $y^* \in \hat{Y}_N(\hat{x})$. Then since $\text{dist}(Y_N, Y) \to 0$ as $N \to \infty$, there exists a $y'_N \in Y_N$ such that $y'_N \to y^*$. Hence,

$$\liminf_{N\to\infty} \psi_N(x_N) \geq \lim_{N\to\infty} \omega(x_N, y'_N) = \omega(\hat{x}, y^*) = \psi(\hat{x}). \tag{IV.12}$$

This proves that if $x_N \to \hat{x}$ as $N \to \infty$, then $\psi_N(x_N) \to \psi(\hat{x})$. Based on Proposition I.3, the conclusion follows. □

Lemma IV.6. *Suppose that Assumption IV.2 holds. The sequence of problems* $\{(\text{GMXM}_N)\}_{N\in\mathbb{N}}$ *epi-converges to* (GMXM) *as* $N \to \infty$.

Proof. The proof follows the same arguments as the proof for Lemma IV.5, with Proposition IV.4 replacing Proposition IV.3. □

We next provide two theorems, which directly follow from the epi-convergence of the discretized min-max-min problems to (MXM). Again, we first consider the constant Z case before the variable Z case.

Theorem IV.7. *Suppose that Assumption IV.1 holds. If* $\{\hat{x}_N\}$ *is a sequence of global minimizers of* (SMXM_N) *and there exists an infinite subset* $K \in \mathbb{N}$ *such that* $\hat{x}_N \to^K \hat{x}$ *as* $N \to \infty$, *then* $\hat{x}$ *is a global minimizer of* (SMXM), *and* $\psi_N(\hat{x}_N) \to^K \psi(\hat{x})$ *as* $N \to \infty$.

Proof. The conclusion follows from Lemma IV.5 and Proposition I.4. □

Theorem IV.8. *Suppose that Assumption IV.2 holds. If* $\{\hat{x}_N\}$ *is a sequence of global minimizers of* (GMXM$_N$) *and there exists an infinite subset* $K \in \mathbb{N}$ *such that* $\hat{x}_N \to^K \hat{x}$ *as* $N \to \infty$*, then* $\hat{x}$ *is a global minimizer of* (GMXM)*, and* $\psi_N(\hat{x}_N) \to^K \psi(\hat{x})$ *as* $N \to \infty$*.*

Proof. The conclusion follows from Lemma IV.6 and Proposition I.4. □

Theorem IV.7 imply that if we pick a large $N \in \mathbb{N}$, and solve (SMXM$_N$) to obtain a global minimizer $\hat{x}_N$, then $\hat{x}_N$ is an approximation to a global minimizer of (SMXM). The same is true regarding Theorem IV.8. As (SMXM$_N$) and (GMXM$_N$) are still difficult problems to solve, we reformulate them into finite minimax problems next.

b. Equivalent Finite Minimax Problem

In this subsection, we first introduce a discretized min-min-max problem that we show is equivalent to the discretized min-max-min (GMXM$_N$) in some sense. We then show that the new min-min-max problem can be seen as a finite minimax problem. To show the equivalence of this new min-min-max problem to (GMXM$_N$), we introduce some notational changes to (GMXM$_N$).

Without loss of generality, we assume that $Y_N = \{y_1, y_2, ..., y_N\}$, and we re-express

$$\psi_N(x) = \max_{j \in \mathcal{N}} \min_{z \in Z(x, y_j)} \varphi^j(x, z), \tag{IV.13}$$

where $\mathcal{N} \triangleq \{1, 2, ..., N\}$, and the function $\varphi^j : \mathbb{R}^d \times \mathbb{R}^s \to \mathbb{R}, j \in \mathcal{N}$, is defined by

$$\varphi^j(x, z) \triangleq \phi(x, y_j, z). \tag{IV.14}$$

We now introduce the equivalent problem to (GMXM$_N$):

$$(\text{GMMX}_N) \quad \min_{x \in X} \bar{\psi}_N(x), \tag{IV.15}$$

where $\bar{\psi}_N : \mathbb{R}^d \to \mathbb{R}, N \in \mathbb{N}$, is defined by

$$\bar{\psi}_N(x) \stackrel{\triangle}{=} \min_{\bar{z} \in Z^N(x)} \max_{j \in \mathcal{N}} \bar{\varphi}^j(x, \bar{z}), \tag{IV.16}$$

$\bar{\varphi}^j : \mathbb{R}^d \times \mathbb{R}^{Ns} \to \mathbb{R}, j \in \mathcal{N}$, is defined by

$$\bar{\varphi}^j(x, \bar{z}) \stackrel{\triangle}{=} \varphi^j(x, z_j), \tag{IV.17}$$

$\bar{z} \stackrel{\triangle}{=} (z_1^T, z_2^T, ..., z_N^T)^T$, $z_j \in Z(x, y_j)$ for all $j \in \mathcal{N}$, and $Z^N(x) \stackrel{\triangle}{=} Z(x, y_1) \times Z(x, y_2) \times ... \times Z(x, y_N)$. In order to allow for the exchange of the min and max operators, we introduce a z variable for each $y \in Y_N$. This expands the dimension of z by a factor of N. We denote the equivalent discretized min-min-max problem for (SMXM) by (SMMX_N).

The next result proves the equivalence of the new discretized min-min-max problem to the discretized min-max-min problem. We first consider the constant Z case. We define $Z^N \stackrel{\triangle}{=} Z \times Z \times ... \times Z$.

Theorem IV.9. *Suppose that Assumption IV.1 holds. Then for all $N \in \mathbb{N}$ and $x \in X$, $\bar{\psi}_N(x) = \psi_N(x)$.*

Proof. For all $x \in X$ and $j \in \mathcal{N}$, since $\varphi^j(x, \cdot)$ is continuous on Z and Z is compact, there exists a $z_j(x) \in Z$ such that

$$\varphi^j(x, z_j(x)) = \min_{z \in Z} \varphi^j(x, z). \tag{IV.18}$$

We define $\bar{z}(x) \stackrel{\triangle}{=} (z_1(x)^T, z_2(x)^T, ..., z_N(x)^T)^T$. Then

$$\begin{aligned} \psi_N(x) &= \max_{j \in \mathcal{N}} \varphi^j(x, z_j(x)) \\ &= \min_{\bar{z} \in Z^N, \bar{z} = \bar{z}(x)} \max_{j \in \mathcal{N}} \bar{\varphi}^j(x, \bar{z}) \\ &\geq \min_{\bar{z} \in Z^N} \max_{j \in \mathcal{N}} \bar{\varphi}^j(x, \bar{z}) = \bar{\psi}_N(x). \end{aligned} \tag{IV.19}$$

Next, for all $x \in X$, since $\max_{j \in \mathcal{N}} \bar{\varphi}^j(x, \cdot)$ is continuous on Z^N, and Z^N is compact, there exists a $\bar{z}(x) \in Z^N$ such that

$$\bar{\psi}_N(x) = \max_{j \in \mathcal{N}} \bar{\varphi}^j(x, \bar{z}(x)). \tag{IV.20}$$

Then

$$\begin{aligned}\bar{\psi}_N(x) &= \max_{j\in\mathcal{N}} \varphi^j(x, z_j(x)) \\ &= \max_{j\in\mathcal{N}} \min_{z=z_j(x), z\in Z} \varphi^j(x, z) \\ &\geq \max_{j\in\mathcal{N}} \min_{z\in Z} \varphi^j(x, z) = \psi_N(x). \end{aligned} \tag{IV.21}$$

The conclusion follows since $\bar{\psi}_N(x) \geq \psi_N(x)$ and $\psi_N(x) \geq \bar{\psi}_N(x)$. □

We observe that the min-min-max problem (SMMX$_N$) is a constrained finite minimax problem of the form

$$(\text{FMX}_N) \quad \min_{w\in W} \Psi_N(w), \tag{IV.22}$$

where

$$\Psi_N(w) \stackrel{\triangle}{=} \max_{j\in\mathcal{N}} f^j(w), \tag{IV.23}$$

$$f^j(w) \stackrel{\triangle}{=} \bar{\varphi}^j(x, \bar{z}), \tag{IV.24}$$

and $w \stackrel{\triangle}{=} (x^T, z_1^T, z_2^T, ..., z_N^T)^T \subset W \stackrel{\triangle}{=} X \times Z^N$.

Note that we obtain the simpler finite minimax problem (FMX$_N$) from the discretized min-max-min (SMXM$_N$) at the expense of a larger number of variables, i.e., $w \in \mathbb{R}^{d+Ns}$.

The results above are next generalized to the variable Z case.

Theorem IV.10. *Suppose that Assumption IV.2 holds. Then for all $N \in \mathbb{N}$ and $x \in X$, $\bar{\psi}_N(x) = \psi_N(x)$.*

Proof. The proof follows the same arguments as the proof for Theorem IV.9, with obvious notational changes. □

The generalized min-min-max problem (GMMX$_N$) is also a constrained finite minimax problem, with a form similar to (FMX$_N$) defined in (IV.22)-(IV.24), except that the set W is replaced by $W_x \stackrel{\triangle}{=} \{(x, \bar{z}) \in \mathbb{R}^d \times \mathbb{R}^{Ns} \mid x \in X, \bar{z} \in Z^N(x)\}$. We denote the constrained finite minimax problem for (GMXM) as (GFMX$_N$).

We next propose an algorithm that produces an approximation to a global minimizer of (MXM) by solving the constructed finite minimax problem.

2. Algorithm for Semi-Infinite Min-Max-Min

In this subsection, under the assumption that there exists a constrained finite minimax algorithm that produces a global minimizer of (FMX_N), we propose an algorithm that obtains a point that is close to a global minimizer of (SMXM). We describe a constrained finite minimax algorithm that satisfies the assumption in the numerical section. In this subsection, we only consider the constant Z case, however, all the results equally apply to the variable Z case.

From this point on, we refer to those algorithms that are applied to solve (FMX_N), $N \in \mathbb{N}$, as *algorithm maps*, to differentiate them from the overall algorithm for (SMXM). We develop the convergence results of our approach based on a constrained finite minimax algorithm map that satisfies the following assumption.

Assumption IV.11. *Suppose that Assumption IV.1 holds. Given an $N \in \mathbb{N}$, the algorithm map applied to solve* (FMX_N) *generates a sequence $\{w_i\}_{i=0}^{\infty} \subset X \times Z^N$, and every accumulation point of that sequence is a global minimizer of* (FMX_N). □

In view of the above results, the following algorithm for (SMXM) is simple.

Algorithm IV.1. Semi-Infinite Min-Max-Min Algorithm

Parameter: $N \in \mathbb{N}$.

Step 1. Generate a sequence $\{w_i\}_{i=0}^{\infty}$ by applying a constrained finite minimax algorithm map that satisfies Assumption IV.11 to (FMX_N). □

The next theorem implies that if we choose a high level of discretization, i.e., large N, then from every accumulation point of the sequence generated, we can easily construct a point that is a global minimizer of the discretized min-max-min problem (SMXM_N). Thus, if the level of discretization N increases to infinity, the points constructed approach the global minimizer of the original semi-infinite min-max-min problem (SMXM), due to Theorem IV.7.

Theorem IV.12. *Suppose that Assumption IV.1 holds, and that Algorithm IV.1 is applied to solve* (SMXM) *with a given $N \in \mathbb{N}$, and it generates a sequence $\{w_i\}_{i=0}^{\infty} \subset$*

$X \times Z^N$. *If* $w^* = (x^*, \bar{z}^*)$ *with* $x^* \in X$ *and* $\bar{z}^* \in Z^N$, *is an accumulation point of* $\{w_i\}_{i=0}^{\infty}$, *then* x^* *is a global minimizer of* (SMXM$_N$).

Proof. The conclusion follows directly from Theorem IV.9. □

D. NUMERICAL RESULTS

In this section, we apply our approach on a DAD problem with a ten-node 18-arc network as shown in Figure 2. The problem parameters, e.g., $ubsupply_u$, $demand_u$, and *totalsorties* are obtained by uniform random number generators based on bounds that we provide. We set the bounds in such a way that more supply can be placed at nodes 1-5 than 6-10, while the demands are higher at nodes 6-10 than 1-5. This ensures that we have flow from the left-hand side of the network to the right-hand side. We refer to Appendix E for the problem parameters generated and used in this study. We use a discretization level of N =1,000, i.e., we consider 1,000 randomly-generated attack plans. Each attack plan provides the sorties to launch against the 18 arcs.

We solve the constrained finite minimax problem constructed in our approach by reformulating it into a standard nonlinear constrained problem and solving it using a sequential quadratic program (SQP) algorithm. We implement and run the algorithm in MATLAB version 7.10 (R2010a) (see Mathworks, 2009) on a 3.46 GHz PC with two quad-core processors, using Windows 7 Pro, with 24 GB of RAM. We use the SQP algorithm in TOMLAB SNOPT solver, see Gill et al. (2007).

For our problem with ten nodes and 18 arcs, and a discretization level $N = 1,000$, (FMX$_N$) has 1,000 functions and approximately 18,000 variables, and takes approximately 4.5 hours, while the (GFMX$_N$)) has 11,000 functions and approximately 38,000 variables, and takes approximately 1.5 hours. The smaller (FMX$_N$) requires a longer run time because there are more nonlinear components in its formulation, where the balance of flow and capacity constraints have been modeled as nonlinear penalty cost terms in the objective function.

We first discuss the results for (SMXM). We refer to Figure 2 for the solutions obtained from solving (FMX_N). The optimal supply solution and the required demand are stated on the nodes. The supply numbers highlighted in red (specifically those for nodes 6-10) indicate that the proposed supplies are at their $ubsupply_u$ values. The worst-case attack plan is one that concentrates attack on arcs (4,6) and (5,7), see the details on this attack plan in Appendix E. This worst-case attack plan is reasonable based on the problem parameters, where more supply can be placed at those nodes on the left-hand side of the network, while higher demands are required at the nodes on the right-hand side. The optimal flow after the worst-case attack is stated on the arcs, and the objective function value is 1,288.

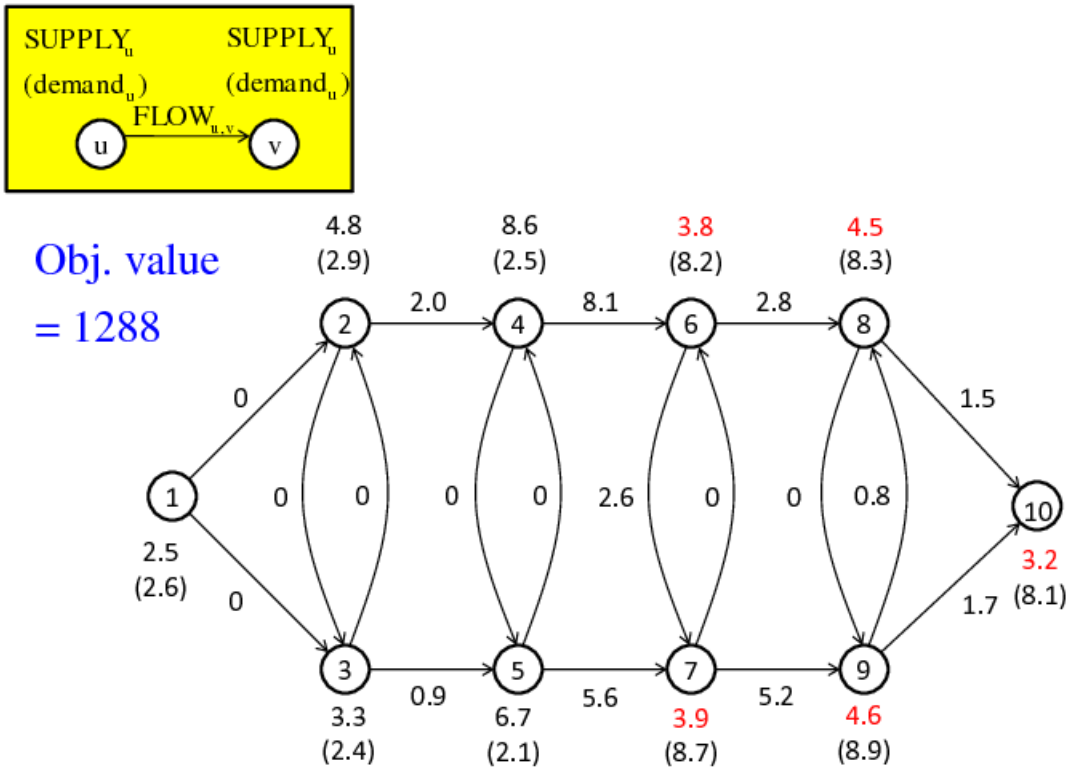

Figure 2. Optimal Supply and Flow Solution for (SMXM).

The proposed supply sums up to 46.0. This is less than the total demand of 54.5. We develop another optimization model, which we refer to as the *verification*

model, to verify if the solution obtained from our approach is reasonable, and to determine why the proposed supply is less than the total demand. When given an arbitrary *SUPPLY*, the verification model runs through all 1,000 attack plans, and for each attack plan, determines the optimal flow and associated objective function value. We test the verification model with several alternative supply solutions that sum to the required demand of 54.5, and obtain objective function values no smaller than approximately 2,300. We conclude that the proposed supply is less than the total demand because the sorties have reduced the capacity of the network to an extent that any additional supply is unable to flow to satisfy any outstanding demand. Thus, we do not gain any benefit by adding supply, and worse still, we incur the additional cost of storing supply as well as incur penalty for balance of flow constraint violations as the additional supply cannot flow out.

We next state the results for (GMXM). We refer to Figure 3 for the solutions obtained from solving (GFMX_N). The proposed supply is the same as that proposed for (SMXM), except for the smaller supply placed at nodes 1, 4, and 5. The optimal flow after the worst-case attack is stated on the arcs, and the objective function value is 891, see the details on the attack plan in Appendix E. The objective function value for (GMXM) is significantly different from that of (SMXM) as the two problems have different objective functions. The worst-case attack and the optimal flow for (GMXM) are significantly different from that of (SMXM).

E. CONCLUSIONS FOR SEMI-INFINITE MIN-MAX-MIN

This chapter focuses on the semi-infinite min-max-min problem. We propose an approach that constructs a finite minimax problem with a larger dimensionality than the original min-max-min problem, through discretization and reformulation of the original problem. Our approach is the first to solve the generalized semi-infinite min-max-min problem, and it also solves the semi-infinite min-max-min problem. The numerical results show that the approach produces reasonable solutions.

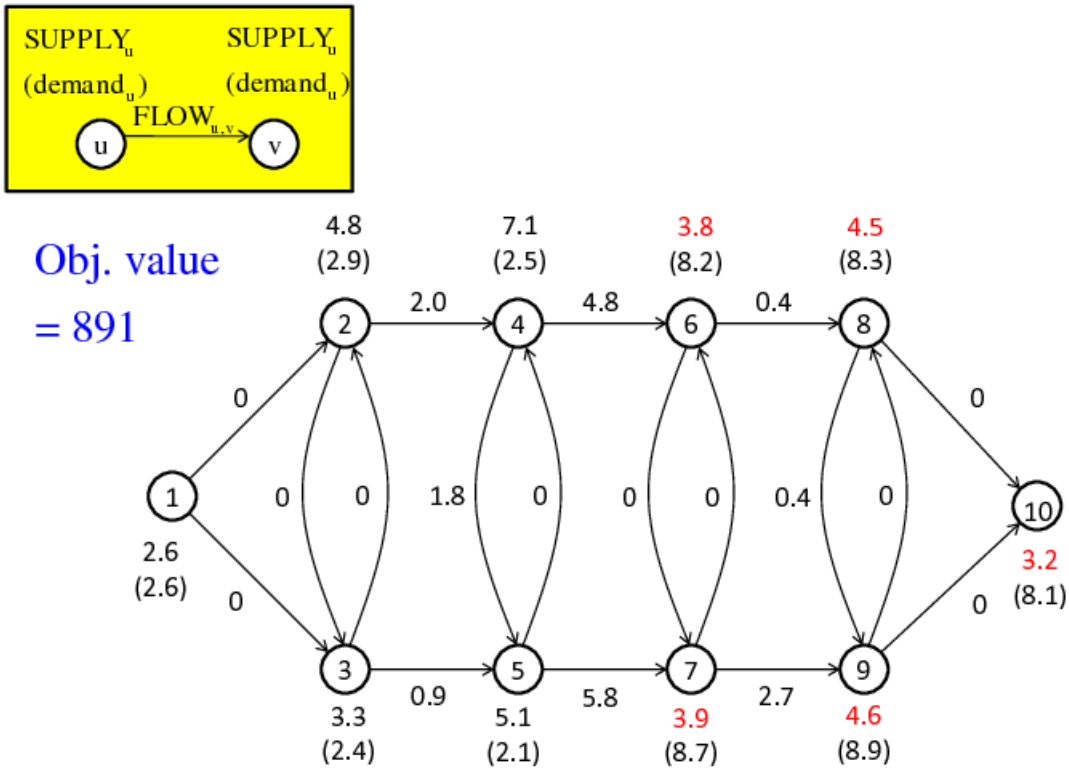

Figure 3. Optimal Supply and Flow Solution for (GMXM).

V. CONCLUSIONS AND FUTURE RESEARCH

A. CONCLUSIONS

Optimization problems with uncertain parameters arise in numerous applications. One possible approach to handle such problems is to consider the worst-case value of the uncertain parameter during optimization. We consider three problems resulting from this approach: a finite minimax problem (FMX), a semi-infinite minimax problem (SMX), and a semi-infinite min-max-min problem (MXM). In all problems, we consider nonlinear functions with continuous variables. We develop rate of convergence and complexity results, and propose algorithms for solving these optimization problems.

We develop rate of convergence and complexity results of smoothing algorithms for solving (FMX) with many functions. We find that smoothing algorithms may only have sublinear rates of convergence, but their complexity in the number of functions q is $O(q \log q)$, as compared to $O(q^3)$ for the sequential quadratic programming (SQP) algorithms, which our numerical results as well as those in the literature show to be one of the fastest for solving (FMX). The competitive complexity for smoothing algorithms is due to its small computational work per iteration. We present two new smoothing algorithms for (FMX) with novel precision-adjustment schemes, and show that they are competitive with other algorithms from the literature. They are especially efficient for problems with many variables, or where a significant number of functions are nearly active at stationary points. The new algorithms are easy to implement and do not require any QP solver, which is required for algorithms such as the SQP and Pshenichnyi-Pironneau-Polak (PPP) minimax algorithm. One of our proposed precision-adjustment schemes is simpler and more efficient than the scheme used in the existing smoothing algorithms, which provides a good alternative when developing new smoothing algorithms. Our numerical results indicate that smoothing

with first-order gradient methods is likely the only viable approach to solve a (FMX) with a large number q of functions and problem dimensionality d, due to memory limitations. The SQP and PPP algorithms need to compute and provide the gradient information ($q \times d$ matrix) to the QP solver, and so the size of the problem that can be solved is limited by the memory required to store the $q \times d$ matrix, as well as the memory required by the QP solver to process the gradient information. In smoothing algorithms, we do not require the memory to store the full $q \times d$ matrix, as the gradient of the smoothed function can be constructed by sequentially considering portions of the gradient matrix.

For (SMX), we develop and compare rate of convergence results for various fixed and adaptive discretization algorithms, as well as an ϵ-subgradient algorithm. We present a novel way of expressing rate of convergence, in terms of computational work instead of the typical number of iterations, which we use throughout the analysis of (SMX). Hence, we are able to identify algorithms that are competitive due to low computational work per iteration even if they require many iterations. We show that to solve (SMX), a fixed discretization algorithm with quadratically or linearly convergent algorithm map to solve the discretized problem can achieve the same asymptotic convergence rate attained by an adaptive discretization method. Under certain convexity-concavity assumptions, we show how the rate of convergence for discretization algorithms depend on the dimension of the uncertain parameters, while ϵ-subgradient algorithms do not. This indicates that, under convexity-concavity assumptions, discretization algorithms will not be competitive against ϵ-subgradient algorithms for moderate to large dimension of the uncertain parameters. Our numerical results show that discretization algorithms are not competitive to ϵ-subgradient algorithms for convex-concave problems with a dimension of the uncertain parameters as small as two.

We propose a new approach to solve (MXM), based on discretization and reformulation of (MXM) into a constrained finite minimax problem with a larger dimen-

sionality than the original (MXM). Our approach is the first to solve (GMXM) in the literature, and it also solves (SMXM). We apply our approach to a defender-attacker-defender network interdiction problem, and the results demonstrate the viability of our approach.

B. FUTURE RESEARCH

There are several possibilities for extending the research of this dissertation. The two smoothing algorithms developed for (FMX) in this dissertation produce a working set that is monotonically increasing. The efficiency of the active-set SQP algorithm shows the potential benefits of an aggressive active-set strategy that keeps the working set small. However, when we implement the active-set strategy from the SQP algorithm in our smoothing algorithms, we see slower run times, which indicates that some kind of fine-tuning on the active-set strategy is probably required. Thus, an extension would be to custom-fit an active-set strategy for smoothing algorithms.

Another opportunity for extension concerns the precision-adjustment scheme in the smoothing algorithm for (FMX) that requires user-specified parameters. It would be worthwhile to develop procedures for rationally selecting these parameters, as it is difficult for users to come up with good choices for the parameters.

We show that the ϵ-subgradient algorithm has better rate of convergence for solving (SMX) than discretization algorithm. However, the ϵ-subgradient algorithm requires a concavity assumption to ensure that the computational work to obtain an ϵ-maximizer (global maximum) for the uncertain parameters remains bounded. Without the concavity assumption, it would be interesting to see how the rate of convergence results for other algorithms such as the exchange algorithms compare to discretization algorithms, since exchange algorithms will also need to implement some form of discretization or branch-and-bound techniques to obtain a global maximizer for the uncertain parameters.

Our approach to solve (MXM) constructs a constrained finite minimax problem

with a large number of functions and variables. The constructed problem has a special structure, each function depends on only a small number of variables, the same number as the sum of the number of variables in the innermost and outermost minimization problem. It would be useful to develop special first-order algorithms that utilize this special structure.

APPENDIX A. FINITE MINIMAX PROBLEMS

Table 17 describes the problem instances used for the numerical studies in Chapter II. Most columns are self-explanatory. Columns 2 and 3 give the number of variables d and functions q, respectively. The target values (Column 7) are equal to the optimal values (if known) or a slightly adjusted value from the optimal values reported in Polak et al. (2003); Zhou and Tits (1996) for smaller q. The same target values are used for ProbA-ProbM in Tables 5 and 6.

In this appendix, we denote components of $x \in \mathbb{R}^d$ by subscripts, i.e., $x = (x_1, x_2, ..., x_d) \in \mathbb{R}^d$. When the problem is given in semi-infinite form, as in (A.2a) - (A.2i), the set Y is discretized into q equally spaced points if

$$\psi(x) = \max_{y \in Y} \phi(x, y), \tag{A.1a}$$

and $q/2$ equally spaced points if

$$\psi(x) = \max_{y \in Y} |\phi(x, y)|. \tag{A.1b}$$

ProbA is defined by (A.1a) and (A.2a), and ProbB-ProbI by (A.1b) and (A.2b)-(A.2i), respectively.

$$\phi(x, y) = (2y^2 - 1)x + y(1 - y)(1 - x), \quad Y = [0, 1], \tag{A.2a}$$

$$\phi(x, y) = (1 - y^2) - (0.5x^2 - 2yx), \quad Y = [-1, 1], \tag{A.2b}$$

$$\phi(x, y) = y^2 - (yx_1 + x_2 \exp(y)), \quad Y = [0, 2], \tag{A.2c}$$

$$\phi(x, y) = \frac{1}{1 + y} - x_1 \exp(yx_2), \quad Y = [-0.5, 0.5], \tag{A.2d}$$

$$\phi(x, y) = \sin y - (y^2 x_3 + yx_2 + x_1), \quad Y = [0, 1], \tag{A.2e}$$

$$\phi(x, y) = \exp(y) - \frac{x_1 + yx_2}{1 + yx_3}, \quad Y = [0, 1], \tag{A.2f}$$

$$\phi(x,y) = \sqrt{y} - [x_4 - (y^2 x_1 + y x_2 + x_3)^2], \quad Y = [0.25, 1], \tag{A.2g}$$

$$\phi(x,y) = \frac{1}{1+y} - [x_1 \exp(y x_3) + x_2 \exp(y x_4)], \quad Y = [-0.5, 0.5], \tag{A.2h}$$

$$\phi(x,y) = \frac{1}{1+y} - [x_1 \exp(y x_4) + x_2 \exp(y x_5) + x_3 \exp(y x_6)],$$
$$Y = [-0.5, 0.5], \tag{A.2i}$$

ProbJ-ProbM are defined by $\psi(x) = \max_{j \in Q} f^j(x)$, with $f^j(x)$ as in (A.2j)-(A.2m), respectively.

$$f^j(x) = x_j^2, \quad j = \{1, ..., q\}, \tag{A.2j}$$

$$f^j(x) = x_{(j-1)2+1}^2 + x_{2j}^2, \quad j = \{1, ..., q\}, \tag{A.2k}$$

$$f^j(x) = x_{(j-1)4+1}^2 + x_{(j-1)4+2}^2 + x_{(j-1)4+3}^2 + x_{4j}^2, \quad j = \{1, ..., q\}, \tag{A.2l}$$

$$f^j(x) = x_{k_j}^2 + x_{l_j}^2, \quad j = \left\{1, 2, 3, ..., \binom{d}{2}\right\}, \tag{A.2m}$$

where (k_j, l_j) are all 2-combinations (see Section 3.3 of Brualdi 2004) of $\{1, 2, 3, ..., d\}$, and

$$f^j(x) = a_j x_i^2 + b_j x_i + c_j, j = \{1, ..., q\}, \tag{A.2n}$$

where $i = \left\lceil \frac{j}{q/d} \right\rceil$, and a_j, b_j, c_j are randomly generated from a uniform distribution on $[0.5, 1]$.

Instance	d	q	$\psi(x)$	Convexity	Initial point	Target value	Ref.
ProbA	1	varies	(A.1a), (A.2a)	Convex	5	0.1783942	Polak et al. (2003)
ProbB	1	varies	(A.1b), (A.2b)	Non-convex	1	1.0000100	Zhou and Tits (1996)
ProbC	2	varies	(A.1b), (A.2c)	Convex	$(1, 1)$	0.5382431	Zhou and Tits (1996)
ProbD	2	varies	(A.1b), (A.2d)	Non-convex	$(1, -1)$	0.0871534	Zhou and Tits (1996)
ProbE	3	varies	(A.1b), (A.2e)	Convex	$(1, 1, 1)$	0.0045048	Polak et al. (2003)
ProbF	3	varies	(A.1b), (A.2f)	Non-convex	$(1, 1, 1)$	0.0042946	Zhou and Tits (1996)
ProbG	4	varies	(A.1b), (A.2g)	Non-convex	$(1, 1, 1, 1)$	0.0026500	Polak et al. (2003)
ProbH	4	varies	(A.1b), (A.2h)	Non-convex	$(1, 1, -3, -1)$	0.0020688	Zhou and Tits (1996)
ProbI	6	varies	(A.1b), (A.2i)	Non-convex	$(1, 1, 1, -7, -3, -1)$	0.0006242	Zhou and Tits (1996)
ProbJ	q	varies	(II.2), (A.2j)	Convex	$(\frac{2}{q}, \frac{4}{q}, \frac{6}{q}, ..., 1, -1 - \frac{2}{q}, ..., -2)$	0	Polak et al. (2003)
ProbK	$2q$	varies	(II.2), (A.2k)	Convex	$(\frac{1}{q}, \frac{2}{q}, \frac{3}{q}, ..., 1, -1 - \frac{1}{q}, ..., -2)$	0	Polak et al. (2003)
ProbL	$4q$	varies	(II.2), (A.2l)	Convex	$(\frac{1}{2q}, \frac{2}{2q}, \frac{3}{2q}, ..., 1, -1 - \frac{1}{2q}, ..., -2)$	0	Polak et al. (2003)
ProbM	varies	$\binom{d}{2}$	(II.2), (A.2m)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0	*
ProbN(i)	10	10,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9278640	*
ProbN(ii)	100	10,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9313887	*
ProbN(iii)	1,000	10,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9288089	*
ProbN(iv)	10	100,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9307828	*
ProbN(v)	100	100,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9340950	*
ProbN(vi)	1,000	100,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9366594	*
ProbN(vii)	1,000	1,000,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9358776	*
ProbN(viii)	1,000	10,000,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9369501	*
ProbN(ix)	10,000	100,000	(II.2), (A.2n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9335266	*

Table 17. Finite minimax problem instances. An asterisk * indicates that the problem instance are created by the authors.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. FINITE MINIMAX ALGORITHM DETAILS AND PARAMETERS

PPP. Pshenichnyi-Pironneau-Polak min-max algorithm (Algorithm 2.4.1 in Polak 1997) use $\alpha = 0.5, \beta = 0.8$, and $\delta = 1$. We use the same Armijo parameters α and β for all algorithms.

ϵ-PPP. ϵ-Active PPP algorithm (Algorithm 2.4.34 in Polak (1997); see also Polak 2008) use the same parameters as above. We implement the most recent version Polak (2008).

SQP-2QP. Sequential Quadratic Programming with two QPs in each iteration (Algorithm 2.1 of Zhou & Tits 1996) use parameters recommended in Zhou and Tits (1996) and monotone line search. (We examined the use of nonmonotone line search in CFSQP, but find it inferior to monotone line search on the set of problem instances.)

SQP-1QP. Sequential Quadratic Programming with one QP in each iteration (Algorithm A in Zhu et al. 2009) use mid-point values stated in Algorithm A, $\alpha = 0.25$ (not the Armijo parameter), $\tau = 2.5$, and $H_0 = I$. The same settings for α and H_0 are used by a co-author in Zhu and Zhang (2005).

SMQN. Smoothing Quasi-Newton algorithm (Algorithm 3.2 in Polak et al. 2008) use $p_0 = 1$, $B(\cdot) = I$, and Parameter Adjustment subroutine version "Case (A)" of Polak et al. (2003).

Algorithm II.2. This algorithm uses the same parameters as SMQN, except for in the Adaptive Penalty Parameter Adjustment subroutine, where it uses $\xi = 2, \varsigma = 2$.

Algorithm II.3. This algorithm use parameters $t = 10^{-5}, \varphi = 1, p_0 = 1, \hat{p} = (\log q/t) \cdot 10^{10}, \kappa = 10^{30}, \xi = 2, \gamma = t \cdot 10^{-10}, \nu = 0.5, \Delta p = 10$.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SEMI-INFINITE MINIMAX PROBLEMS

In this appendix, we denote components of $x \in \mathbb{R}^d$ and $y \in Y \subset \mathbb{R}^m$ by subscripts, for example, $x = (x_1, x_2, ..., x_d)$. Two problem instances from Rustem and Howe (2002) are used for the numerical studies in Chapter III. The problem (SMX) to be solved is as defined by (III.1) and (III.2), with $\phi(x, y)$ as defined below:

$$\phi(x,y) = 5\sum_{i=1}^{2} x_i^2 - y^2 + x_1(-y+5) + x_2(y+3), \tag{C.1a}$$

$$\phi(x,y) = 5\sum_{i=1}^{2} x_i^2 - \sum_{i=1}^{2} y_i^2 + x_1(-y_1+y_2+5) + x_2(y_1-y_2+3), \tag{C.1b}$$

$$\begin{aligned}\phi(x,y) &= -(x_1-1)y_1 - (x_2-2)y_2 - (x_3-1)y_3 + 2x_1^2 + 3x_2^2 + x_3^2 \\ &- \sum_{i=1}^{3} y_i^2.\end{aligned} \tag{C.1c}$$

The second (SProbB) and third (SProbC) problem instances are Problems 1 and 5 on pp. 100-102 of Rustem and Howe (2002), respectively. SProbB and SProbC have y-dimensionality of two and three respectively. There are no problem instances in Rustem and Howe (2002) with y-dimensionality of one. We create SProbA from SProbB by removing y_2 and replacing y_1 by y. All three problem instances are convex-concave, i.e., $\phi(\cdot, y)$ is convex for any fixed $y \in Y$, and $\phi(x, \cdot)$ is concave for any fixed $x \in \mathbb{R}^d$. As $\phi(\cdot, y)$ is convex for any fixed y, a subgradient is guaranteed to exist, which is a pre-requisite for the ϵ-subgradient algorithm, Algorithm III.3. As $\phi(x, \cdot)$ is strictly concave for any fixed x, there exists a unique y-maximizer for each fixed x.

Table 18 provides more details on the problem instances. Columns 2 and 3 give the dimensions of the solution space d and the uncertain parameter m, respectively. Columns 4 and 5 give the initial points to the solution x_0 and the uncertain parameter y_0, respectively. Note that y_0 is only relevant for the ϵ-subgradient algorithm, Algorithm III.3, as it is not required for the discretization algorithms.

For the target solutions (last column), we use Algorithm III.3 (as it shows significantly faster run times than the discretization algorithms during the preliminary experiments) with parameters as in Appendix D, we start with a stepsize of $\alpha = 0.1$ and run the algorithm until the solution remains unchanged for more than ten iterations, we use this solution to warm-start the next stage where we decrease the stepsize to $\alpha = 0.01$ and repeat the process until $\alpha = 10^{-5}$. We do not use the optimal solutions as reported in Rustem and Howe (2002) as Rustem and Howe (2002) uses a different termination criteria. The optimal solutions obtained with our procedure agree with those reported in Rustem and Howe (2002) at least to the fourth decimal place.

The other columns are self-explanatory.

Instance	d	m	$\phi(x,y)$	x_0	y_0	Bounds on y	Target solution
SProbA	2	1	(C.1a)	(10,-10)	5	$-5 \leq y \leq 5$	(-0.49090909, -0.30909091)
SProbB	2	2	(C.1b)	(10,-10)	(5, -5)	$-5 \leq y_1, y_2 \leq 5$	(-0.48333332, -0.31666667)
SProbC	3	3	(C.1c)	(2, 2, 2)	(1, 1, 1)	$-1 \leq y_1, y_2, y_3 \leq 1$	(0.11111111, 0.15384615, 0.20000000)

Table 18. Semi-infinite minimax problem instances. The starting value y_0 is only relevant for the ϵ-subgradient algorithm, Algorithm III.3.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. SEMI-INFINITE MINIMAX ALGORITHM DETAILS AND PARAMETERS

Algorithm III.1 (applies to both ϵ-PPP and SQP-2QP). Given a discretization parameter $N \in \mathbb{N}$ ($|Y_N|$), the discretization scheme discretize each dimension of y into $N^{1/m}$ equally spaced points, which gives a total of N grid points. For the numerical studies, N are chosen such that $N^{1/m}$ are integers.

Algorithm III.1 with ϵ-PPP. The ϵ-PPP algorithm is the same algorithm used in Chapter II. The same Armijo parameters $\alpha = 0.5, \beta = 0.8$, and $\delta = 1$ are used. The ϵ parameter for determining the active set is set at 10^{-3}, which is the value used for the algorithm comparison in Chapter II.

Algorithm III.1 with SQP-2QP. The SQP-2QP algorithm is the same algorithm used in Chapter II. Similar to Chapter II, we use the parameters recommended in Zhou and Tits (1996) and monotone line search. The ϵ parameter for determining the active set is set at 1, which is the value used for the algorithm comparison in Chapter II.

Algorithm III.3. Our preliminary numerical tests show very fast run times for Algorithm III.3 as compared to the other two discretization algorithms. Thus, we spent minimal effort in sensitivity analyses to fine-tune the algorithm parameters. We use a constant stepsize $\alpha = 0.1$, which the preliminary tests show to be robust. For Step 2 of Algorithm III.3, we use TOMLAB SNOPT with its default tolerances to find the y-maximizer, and the final y iterate from the previous major iteration is used to warm-start the search for the y-maximizer in the current major iteration.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. SEMI-INFINITE MIN-MAX-MIN PROBLEM PARAMETERS AND RESULTS

We generate the data for the defender-attacker-defender (DAD) network using Table 19, which are used for both (SMXM) and (GMXM). Once the problem data are generated, see Table 20, we hold them fixed for the analyses. For data ϵ, *penbal*, and *pencap*, which are not data from the problem but are required for the formulation, we do not randomly generate their values. Instead, we fix their values, $\epsilon = 0.001$, *penbal*= 50, and *pencap*= 50.

Data	**Random generators**
costsupply	5×U(2,4), 5×U(10,13)
costflow	U(1,11)
demand	5×U(2,3), 5×U(8,10)
totalsorties	U(20,22)
ubsupply	U(10,12)
ubflow	U(5,15)
vul	U(0.5,2)
vulcap	$0.6 \times ubflow$

Table 19. Random generators used to produce the DAD problem parameters in Table 20. The phrase 5×U(2,4), 5×U(10,13) represents that a total of ten random numbers are generated, the first five are uniformly distributed between two and four, and the last five numbers are uniformly distributed between ten and 13.

We generate 1,000 random attack plans, each attack plan has total sorties over the 18 arcs no greater than 20.7 sorties. Specifically, we generate 18 random numbers, each U(0,20.7/4). If the sum of the 18 numbers is no greater than 20.7, we accept the set as an attack plan. We repeat until we accumulate 1,000 attack plans. The factor "4" in "20.7/4" is chosen empirically.

An initial point $w_0 = (0.5 \times ubsupply, 0.5 \times ubflow, 0.5 \times ubflow, ...)$ is used for (SMXM). An initial point $w_0 = (0.5 \times ubsupply, 0.5 \times ubflow, 0.5 \times ubflow, ...0.5 \times$

$ubflow, 0, 0, ..., 0)$ is used for (GMXM), where the string of 0's is for *EXCESSSUPPLY* and *EXTRASUPPLY*.

Based on the 1,000 attack plans, the (SMXM) solution obtained from solving (FMX_N) is shown in Table 21, which states (i) the optimal supply distribution plan before the attack, (ii) the worst-case attack plan, and (iii) the optimal flow after the worst-case attack. The objective function value for this solution is 1,288.

For (GMXM), the solution is shown in Table 22, with an objective function value of 891. Note that the objective functions for (SMXM) and (GMXM) are different, which explains the significantly different objective function values.

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Arc	(1,2)	(1,3)	(2,3)	(2,4)	(3,2)	(3,5)	(4,5)	(4,6)	(5,4)	(5,7)	(6,7)	(6,8)	(7,6)	(7,9)	(8,9)	(8,10)	(9,8)	(9,10)
Parameters																		
costsupply	2.3	2.4	2.5	3.9	3.7	10.2	12.8	12.2	12.2	10.2	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
costflow	9.6	10.3	10.8	9.6	8.9	6.1	2.8	5.0	2.3	1.3	10.4	4.0	4.0	4.3	5.7	7.5	1.3	9.4
demand	2.6	2.9	2.4	2.5	2.1	8.2	8.7	8.3	8.9	8.1	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
totalsorties	20.7	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
ubsupply	12.0	11.1	11.4	12.0	10.6	3.8	3.9	4.5	4.6	3.2	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
ubflow	6.8	8.6	5.6	10.2	8.4	6.8	7.1	14.1	11.8	9.7	14.1	6.0	12.5	12.4	10.6	6.8	11.0	8.0
vul	0.7	0.8	1.8	0.6	0.9	0.6	1.2	0.5	1.9	0.8	0.6	1.0	1.2	0.7	2.0	1.0	1.0	0.6
vulcap	4.1	5.2	3.3	6.1	5.0	4.1	4.3	8.4	7.1	5.8	8.5	3.6	7.5	7.4	6.4	4.1	6.6	4.8

Table 20. Defender-Attacker-Defender Network Data.

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Arc	(1,2)	(1,3)	(2,3)	(2,4)	(3,2)	(3,5)	(4,5)	(4,6)	(5,4)	(5,7)	(6,7)	(6,8)	(7,6)	(7,9)	(8,9)	(8,10)	(9,8)	(9,10)
Parameters																		
SUPPLY	2.5	4.8	3.3	8.6	6.7	3.8	3.9	4.5	4.6	3.2	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
SORTIE	0.7	0.4	1.8	0.1	0.3	0.4	0.6	4.3	1.0	4.6	1.1	0.7	1.3	0.2	0.4	1.9	0.0	0.4
FLOW	0.0	0.0	0.0	2.0	0.0	0.9	0.0	8.1	0.0	5.6	2.6	2.8	0.0	5.2	0.0	1.5	0.8	1.7

Table 21. Results for (SMXM).

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Arc	(1,2)	(1,3)	(2,3)	(2,4)	(3,2)	(3,5)	(4,5)	(4,6)	(5,4)	(5,7)	(6,7)	(6,8)	(7,6)	(7,9)	(8,9)	(8,10)	(9,8)	(9,10)
Parameters																		
SUPPLY	2.6	4.8	3.3	7.1	5.1	3.8	3.9	4.5	4.6	3.2	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
SORTIE	0.7	0.9	1.9	0.3	0.2	1.7	4.6	0.3	1.1	0.4	1.8	0.1	0.6	0.8	0.5	0.7	0.4	1.1
FLOW	0.0	0.0	0.0	2.0	0.0	0.9	1.8	4.8	0.0	5.8	0.0	0.4	0.0	2.7	0.4	0.0	0.0	0.0

Table 22. Results for (GMXM).

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications*. Englewood Cliffs, NJ: Prentice-Hall.

Ariyawansa, K. A., & Jiang, P. L. (2000). On complexity of the translational-cut algorithm for convex minimax problems. *J. Optimization Theory and Applications*, *107*, 223–243.

Becker, R. G., Dwolatzky, B., Karakitsos, E., & Rustem, B. (1986). The simultaneous use of rival models in policy optimisation. *The Economic Journal*, *96*, 425–448.

Bertsekas, D. P. (1999). *Nonlinear programming*. Belmont, MA: Athena Scientific.

Bertsekas, D. P. (2010). *Convex optimization theory. Supplementary Chapter 6 on convex optimization algorithms*. Nashua, NH: Athena Scientific, May 5, 2010 edition.

Bertsekas, D. P., Nedic, A., & Ozdaglar, A. E. (2003). *Convex analysis and optimization*. Athena Scientific Optimization and Computation Series. Belmont, MA: Athena Scientific.

Brualdi, R. A. (2004). *Introductory combinatorics*. Upper Saddle River, NJ: Prentice-Hall.

Cai, X., Teo, K., Yang, X., & Zhou, X. (2000). Portfolio optimization under a minimax rule. *Management Science*, *46*, 957–972.

Capps, L. R. (1970). A dynamic model for the allocation of airstrikes against a lines-of-communication network. Master's thesis, Naval Postgraduate School, Monterey, CA.

Cardinal, J., & Langerman, S. (2006). Min-max-min geometric facility location problems. In *Proc. of the European workshop on computational geometry (EWCG06)*, pp. 149–152. Delphi.

Chaney, R. W. (1982). A method of centers algorithm for certain minimax problems. *Mathematical Programming*, *22*, 202–226.

Chen, T., & Fan, M. K. H. (1998). On convex formulation of the floorplan area minimization problem. In *ISPD '98: Proceedings of the 1998 international symposium on physical design*, pp. 124–128. New York, NY: ACM.

Demyanov, V. F., & Malozemov, V. N. (1971). On the theory of non-linear min-max problems. *Russian Mathematical Surveys*, *26*, 57–115.

Demyanov, V. F., & Malozemov, V. N. (1974). *Introduction to minimax*. New York, NY: Wiley.

Drezner, Z. (1987). On the complexity of the exchange algorithm for minimax optimization problems. *Mathematical Programming*, *38*, 219–222.

Gill, P. E., Hammarling, S. J., Murray, W., Saunders, M. A., & Wright, M. H. (1986). *User's guide for LSSOL version 1.0: a Fortran package for constrained linear least-squares and convex quadratic programming*. Stanford, CA: Systems Optimization Laboratory - University of Stanford.

Gill, P. E., Murray, W., & Saunders, M. A. (2007). *User's guide for SNOPT version 7: Software for large-scale nonlinear programming*. Stanford, CA: Systems Optimization Laboratory - University of Stanford.

Gill, P. E., Murray, W., & Wright, M. H. (1991). *Numerical linear algebra and optimization*. Redwood, CA: Addison-Wesley.

Hettich, R. (1986). An implementation of a discretization method for semi-infinite programming. *Mathematical Programming*, *34*, 354–361.

Kirjner-Neto, C., & Polak, E. (1998). On the conversion of optimization problems with max-min constraints to standard optimization problems. *SIAM J. Optimization*, *8*, 887–915.

Kiwiel, K. C. (1987). A direct method of linearization for continuous minimax problems. *J. Optimization Theory and Applications*, *55*, 271–287.

Klessig, R., & Polak, E. (1973). A method of feasible directions using function approximations, with applications to min max problems. *J. Mathematical Analysis and Applications*, *41*, 583–602.

Kort, B. W., & Bertsekas, D. P. (1972). A new penalty function algorithm for constrained minimization. *Proceedings 1972 IEEE Conf. Decision and Control*, pp. 343–362.

Kortanek, K. O., & No, H. (1993). A central cutting plane algorithm for convex semi-infinite programming problems. *SIAM J. Optimization*, *3*, 901–918.

Lang, B. (2000). Direct solvers for symmetric eigenvalue problems. In *Modern methods and algorithms of quantum chemistry, NIC series, vol. 3*, edited by Grotendorst, J. Julich, Germany.

Lawrence, C., Zhou, J. L., & Tits, A. L. (1997). User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical report.

Lee, S.-H., & Glynn, P. W. (2003). Computing the distribution function of a conditional expectation via Monte Carlo: Discrete conditioning spaces. *ACM Trans. Model. Comput. Simul.*, *13*, 238–258.

Li, X. (1992). An entropy-based aggregate method for minimax optimization. *Engineering Optimization*, *18*, 277–285.

Li, X. S., & Fang, S. C. (1997). On the entropic regularization method for solving min-max problems with applications. *Mathematical Methods of Operations Research*, *46*, 119–130.

Luksan, L., Matonoha, C., & Vlcek, J. (2005). Primal interior-point method for large sparse minimax optimization. Technical Report 941, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic.

Martin, P. A. S. (2007). Tri-level optimization models to defend critical infrastructure. Master's thesis, Naval Postgraduate School, Monterey, CA.

Mathworks (2009). *MATLAB 7 getting started guide*. Natick, MA: The MathWorks, Inc.

Monteiro, R. D. C., & Adler, I. (1989). Interior path following primal-dual algorithms. Part II: Convex quadratic programming. *Mathematical Programming*, *44*, 43–66.

Nemirovski, A. S., & Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization*. New York: John Wiley.

Nesterov, Y. (1995). Complexity estimates of some cutting plane methods based on the analytic barrier. *Mathematical Programming*, *69*, 149–176.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course (applied optimization)*. Norwell, MA: Kluwer Academic Publishers.

Nesterov, Y., & Vial, J. P. (2004). Augmented self-concordant barriers and nonlinear optimization problems with finite complexity. *Mathematical Programming*, *99*, 149–174.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Heidelberg: Springer.

Nugent, R. O. (1969). The optimum allocation of airstrikes against a transportation network for an exponential damage function. Master's thesis, Naval Postgraduate School, Monterey, CA.

Obasanjo, E., Tzallas-Regas, G., & Rustem, B. (2010). An interior-point algorithm for nonlinear minimax problems. *J. Optimization Theory and Applications*, *144*, 291–318.

Panier, E. R., & Tits, A. L. (1989). A globally convergent algorithm with adaptively refined discretization for semi-infinite optimization problems arising in engineering design. *IEEE Transactions on Automatic Control*, *34*, 903–908.

Panin, V. M. (1981). Linearization method for continuous min-max problems. *Kibernetika*, *2*, 75–78.

Pasupathy, R. (2010). On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Oper. Res.*, *58*, 889–901.

Pee, E. Y., & Royset, J. O. (2010). On solving large-scale finite minimax problems using exponential smoothing. To be published in J. Optimization Theory and Applications, available online DOI: 10.1007/s10957-010-9759-1.

Polak, E. (1987). On the mathematical foundations of nondifferentiable optimization in engineering design. *SIAM Review*, *29*, 21–89.

Polak, E. (1997). *Optimization. algorithms and consistent approximations*. New York, NY: Springer.

Polak, E. (2003). Smoothing techniques for the solution of finite and semi-infinite min-max-min problems. In *High performance algorithms and software for nonlinear optimization*, edited by Pillo, G. D., & Murli, A. Dordrecht, Netherlands: Kluwer Academic Publishers.

Polak, E. (2008). On the convergence of the Pshenichnyi-Pironneau-Polak minimax algorithm with an active-set strategy. *J. Optimization Theory and Applications*, *138*, 305–309.

Polak, E., & He, L. (1992). Rate-preserving discretization strategies for semi-infinite programming and optimal control. *SIAM J. Control and Optimization*, *30*, 548–572.

Polak, E., Mayne, D. Q., & Higgins, J. (1992). On the extension of Newton's method to semi-infinite minimax problems. *SIAM J. Control and Optimization*, *30*, 376–389.

Polak, E., Royset, J. O., & Womersley, R. S. (2003). Algorithms with adaptive smoothing for finite minimax problems. *J. Optimization Theory and Applications*, *119*, 459–484.

Polak, E., Salcudean, S., & Mayne, D. Q. (1987). Adaptive control of ARMA plants using worst case design by semi-infinite optimization. *IEEE Transactions on Automatic Control*, *32*, 388–397.

Polak, E., Womersley, R. S., & Yin, H. X. (2008). An algorithm based on active sets and smoothing for discretized semi-infinite minimax problems. *J. Optimization Theory and Applications*, *138*, 311–328.

Powell, W. (2007). *Approximate dynamic programming: Solving the curses of dimensionality*. New York: Wiley.

Price, C. J., & Coope, I. D. (1990). An exact penalty function algorithm for semi-infinite programmes. *BIT Numerical Mathematics*, *30*, 723–734.

Ralph, D., & Polak, E. (2000). A first-order algorithm for semi-infinite min-max-min problems. Manuscript, Judge Institute of Management, University of Cambridge, Cambridge, U.K.

Reemtsen, R. (1991). Discretization methods for the solution of semi-infinite programming problems. *J. Optimization Theory and Applications*, *71*, 85–103.

Rockafellar, R. T. (2007). Coherent approaches to risk in optimization under uncertainty. *Tutorials in Oper. Res.*, *34*, 38–61.

Rockafellar, R. T., & Wets, R. J. B. (1998). *Variational analysis*. Heidelberg: Springer.

Rustem, B., & Howe, M. (2002). *Algorithms for worst-case design and applications to risk management*. Princeton, NJ: Princeton University Press.

Sahinidis, N. V. (2004). Optimization under uncertainty: State-of-the-art and opportunities. *Computers and Chemical Engineering*, *28*, 971–983. FOCAPO 2003 Special issue.

Shapiro, A. (2009). Semi-infinite programming, duality, discretization and optimality conditions. *Optimization: A Journal of Mathematical Programming and Operations Research*, *58*, 133–161.

Shapiro, A., Dentcheva, D., & Ruszczyński, A. (2009). *Lectures on stochastic programming: Modeling and theory*. Philadelphia, PA: SIAM.

Still, G. (2001). Discretization in semi-infinite programming: the rate of convergence. *Mathematical Programming*, *91*, 53–69.

Sturm, J. F., & Zhang, S. (1995). A dual and interior-point approach to solve convex min-max problems. In *Minimax and applications*, edited by Du, D. Z., & Pardalos, P. M., pp. 69–78. Kluwer Academic Publishers.

Tits, A. L. (1985). On the optimal design centering, tolerancing, and tuning problem. *J. Optimization Theory and Applications*, *45*, 487–494.

Tomlab (2009). *User's guide for TOMLAB/CPLEX v12.1.* Pullman, WA: Tomlab Optimization Inc.

Urruty, H., & Baptiste, J. (1996). *Convex analysis and minimization algorithms 1. Fundamentals*. Berlin: Springer.

Wiest, E. J., & Polak, E. (1991). On the rate of convergence of two minimax algorithms. *J. Optimization Theory and Applications*, *71*, 1–30.

Xu, S. (2001). Smoothing method for minimax problems. *Computational Optimization and Applications*, *20*, 267–279.

Ye, F., Liu, H., Zhou, S., & Liu, S. (2008). A smoothing trust-region Newton-CG method for minimax problem. *Applied Mathematics and Computation*, *199*, 581–589.

Zhou, J. L., & Tits, A. L. (1996). An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM J. Optimization*, *6*, 461–487.

Zhu, Z., Cai, X., & Jian, J. (2009). An improved SQP algorithm for solving minimax problems. *Applied Mathematics Letters*, *22*, 464–469.

Zhu, Z., & Zhang, K. (2005). A superlinearly convergent sequential quadratic programming algorithm for minimax problems. *Chinese J. Numerical Mathematics and Applications*, *27*, 15–32.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Assistant Professor Johannes O. Royset
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

4. Distinguished Professor Gerald G. Brown
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

5. Distinguished Professor R. Kevin Wood
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

6. Associate Professor W. Matthew Carlyle
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

7. Associate Professor Craig W. Rasmussen
   Department of Applied Mathematics
   Naval Postgraduate School
   Monterey, California

8. Professor Wei Kang
   Department of Applied Mathematics
   Naval Postgraduate School
   Monterey, California